Priority Algorithms for the Subset-Sum Problem

Yuli Ye and Allan Borodin

Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
{y3ye,bor}@cs.toronto.edu

Abstract. Greedy algorithms are simple, but their relative power is not well understood. The priority framework [5] captures a key notion of "greediness" in the sense that it processes (in some locally optimal manner) one data item at a time, depending on and only on the current knowledge of the input. This algorithmic model provides a tool to assess the computational power and limitations of greedy algorithms, especially in terms of their approximability. In this paper, we study priority algorithm approximation ratios for the Subset-Sum Problem, focusing on the power of revocable decisions. We first provide a tight bound of $\alpha \approx 0.657$ for irrevocable priority algorithms. We then show that the approximation ratio of fixed order revocable priority algorithms is between $\beta \approx 0.780$ and $\gamma \approx 0.852$, and the ratio of adaptive order revocable priority algorithms is between 0.8 and $\delta \approx 0.893$.

1 Introduction

Greedy algorithms are of great interest because of their simplicity and efficiency. In many cases they produce reasonable (and sometimes optimal) solutions. Surprisingly, it is not obvious how to formalize the concept of a greedy algorithm and given such a formalism how to determine its power and limitations with regard to natural combinatorial optimization problems. Borodin, Nielson and Rackoff [5] suggested the priority model which provides a rigorous framework to analyze greedy-like algorithms. In this framework, they define fixed order and adaptive (order) priority algorithms, both of which capture a key notion of greedy algorithms in the sense that they process one data item at a time. For fixed order priority, the ordering function used to evaluate the priority of a data item is fixed before execution of the algorithm, while for adaptive priority, the ordering function can change during every iteration of the algorithm. By restricting algorithms to this framework, approximability results and limitations [1] for many problems have been obtained; for example, scheduling problems [5, 18], facility

[1] We note that similar to the study of online competitive analysis, negative priority results are in some sense incomparable with hardness of approximation results as there are no explicit complexity considerations as to how a priority algorithm can choose its next item and how it decides what to do with that item. Negative results are derived from the structure of the algorithm.

location and set cover [1], job interval selection (JISP and WJISP) [12], and various graph problems [6, 9]. The original priority framework specified that decisions (being made for the current input item) are irrevocable. Even within this restrictive framework, the gap between the best known algorithm and provable negative remains significant for most problems. Following [10, 4], Horn [12] extended the priority framework to allow revocable acceptances when considering packing problems. That is, input items could be accepted and then later rejected, the only restriction being that a feasible solution is maintained at the end of each iteration. The revocable (decision) priority model is intuitively more powerful and almost as conceptually simple as the irrevocable model and it is perhaps surprising that it is not a more commonly used type of algorithm. Erlebach and Spieksma [10] and independently Bar-Noy et al. [4] provide a simple revocable priority approximation algorithm for the WJISP problem, and Horn [12] formalizes this model and provides an approximation upper bound [2] of $\approx 1/(1.17)$ for the special case of the weighted interval scheduling problem. Moore's [17] optimal "greedy algorithm" for the unweighted throughput maximization problem without release times (i.e. $1||\sum_j \bar{U}_j$ in Graham's scheduling notation) can be implemented as a fixed order revocable priority algorithm. It is not difficult to show that this problem cannot be solved optimally by an irrevocable priority algorithm.

The *Subset-Sum Problem* (SSP) is one the most fundamental NP-complete problems [11], and perhaps the simplest of its kind. Approximation algorithms for SSP have been studied extensively in the literature. The first FPTAS (for the more general knapsack problem) is due to Ibarra and Kim [13], and the best current approximation algorithm is due to Kellerer et al. [15], having time and space complexity $O(min\{\frac{n}{\varepsilon}, n+\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\})$ and $O(n+\frac{1}{\varepsilon})$ respectively. Greedy-like approximation algorithms have also been studied for SSP; an algorithm called greedy but using multiple passes, has approximation ratio 0.75, see [16]. In this paper, we study priority algorithms for SSP. Although in some sense one may consider SSP to be a "solved problem", the problem still presents an interesting challenge for the study of greedy algorithms. We believe the ideas employed for SSP will be applicable to the study of simple algorithms for other (say scheduling) problems which are not well understood, such as the throughput maximization problem (with release times) and some of its more tractable subcases. In particular, can we derive priority approximation algorithms for throughput maximization when all jobs have a fixed processing time (i.e. $1|r_j, p_j = p|\sum_j w_j\bar{U}_j$)? (We note that Horn's [12] $1/(1.17)$ bound also applies to this problem.) Baptiste [3] optimally solves this special case of throughput maximization using a dynamic programming algorithm with time complexity $O(n^7)$. (See also Chuzhoy et al. [8] and Chrobak et al. [7] for additional throughput maximization results.)

In spite of the conceptual simplicity of the SSP problem and the priority framework, there is still a great deal of flexibility in how one can design algorithms, both in terms of the ordering and in terms of which items to accept and

[2] As we are considering maximization problems in this paper, all approximation ratios will be ≤ 1 so that negative results become upper bounds on the ratio.

(for the revocable model) which items to discard in order to fit in a new item. We give a tight bound of $\alpha \approx 0.657$ for irrevocable priority algorithms showing that in this case adaptive ordering does not help. For fixed order revocable algorithms, we can show that the best approximation ratio is between $\beta \approx 0.780$ and $\gamma \approx 0.852$; for adaptive revocable priority algorithms, the best approximation ratio is between 0.8 and $\delta \approx 0.893$. All omitted proofs can be found in [19].

2 Definitions and Notation

We use **bold** font letters to denote sets of data items. For a given set $\mathbf{R}$ of data items, we use $|\mathbf{R}|$ to denote its cardinality and $\|\mathbf{R}\|$, its total weight. For a data item u, we use u to represent the singleton set $\{u\}$ and $2u$, the multi-set $\{u, u\}$; we also use u to represent the weight of u since it is the only attribute. The term u here is an overloaded term, but the meaning will become clear in the actual context. For set operations, we use $\oplus$ to denote set addition, and use $\ominus$ to denote set subtraction.

2.1 The Subset-Sum Problem

Given a set of n data items with positive weights and a capacity c, the *maximization version* of SSP is to find a subset such that the corresponding total weight is maximized without exceeding the capacity c. Without loss of generality, we make two assumptions. First of all, the weights are all scaled to their relative values to the capacity; hence we can use 1 instead of c for the capacity. Secondly, we assume each data item has weight $\in (0, 1]$. An instance of SSP is a set $\mathbf{I} = \{u_1, u_2, \ldots, u_n\}$ of n data items, where the set $\mathbf{I}$ is the *input set*, and $u_1, u_2, \ldots, u_n$ are the *data items*. A *feasible* solution is a subset $\mathbf{B}$ of $\mathbf{I}$ such that $\|\mathbf{B}\| \leq 1$. An *optimal* solution is a feasible solution with maximum weight. Let $\mathcal{A}$ be an algorithm for SSP, we denote $\textbf{ALG}$ the solution achieved by $\mathcal{A}$ and $\textbf{OPT}$, the optimal solution, then the approximation ratio of $\mathcal{A}$ on that instance is denoted by $\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|}$. The *approximation ratio* of $\mathcal{A}$ is the infimum of the set of ratios achieved by $\mathcal{A}$ over all instances of SSP.

2.2 Priority Model

We base our terminology and model on that of [5], and start with the class of fixed order irrevocable priority algorithms for SSP. For a given instance, a *fixed order irrevocable* priority algorithm maintains a feasible solution $\mathbf{B}$ throughout the algorithm. The structure of the algorithm [3] is as follows:

[3] We formalize the allowable (fixed) orderings by saying that the algorithm specifies a total ordering on all possible input items. The items that constitute the actual input set $\mathbf{I}$ will then inherit this ordering. That is, the priority model insists that the ordering satisfies Arrow's Independence of Irrelevant Attributes IIA Axion [2]. For adaptive orderings the algorithm can construct a new IIA ordering based on all the items that it has already seen as well as those items it can deduce are not in the input set.

Fixed Order Irrevocable Priority
Ordering: Determine a total ordering of all possible data items
while **I** is not empty
next := index of the data item in **I** that comes first in the ordering
Decision: Decide whether or not to add u_{next} to **B**, and then remove u_{next} from **I**
end while

An *adaptive irrevocable* priority algorithm is similar to a fixed order one, but instead of looking at a data item according to some pre-determined ordering, the algorithm is allowed to reorder the remaining data items in **I** at each iteration. This gives the algorithm an advantage since now it can take into account the information that has been revealed so far to determine which is the best data item to consider next. The structure of an adaptive irrevocable priority algorithm is described as follows:

Adaptive Irrevocable Priority
while **I** is not empty
Ordering: Determine a total ordering of all possible (remaining) data items
next := index of the data item in **I** that comes first in the ordering
Decision: Decide whether or not to add u_{next} to **B**, and then remove u_{next} from **I**
end while

The above defined priority algorithms are "irrevocable" in the sense that once a data item is admitted to the solution it cannot be removed. We can extend our notion of "fixed order" and "adaptive" to the class of revocable priority algorithms, where revocable decisions on accepted data items are allowed. Accordingly, those algorithms are called *fixed order revocable* and *adaptive revocable* priority algorithms respectively. The extension[4] to revocable acceptances provides additional power; for example, as shown in [14], *online irrevocable* algorithms for SSP cannot achieve any constant bound approximation ratio, while *online revocable* algorithms can achieve a tight approximation ratio of $\frac{\sqrt{5}-1}{2} \approx 0.618$.

2.3 Adversarial Strategy

We utilize an adversary in proving approximation bounds. For a given priority algorithm, we run the adversary against the algorithm in the following scheme. At the beginning of the algorithm, the adversary first presents a set of data items to the algorithm, possibly with some data items having the same weight. Furthermore, our adversary promises that the actual input is contained in this set[5]. Since weight is the only input parameter, the algorithm give the same

[4] This extension applies to priority algorithms for packing problems.
[5] This assumption is optional. The approximation bounds clearly hold for a stronger adversary.

priority to all items having the same weight [6]. At each step, the adversary asks the algorithm to select one data item in the remaining set and make a decision on that data item. Once the algorithm makes a decision on the selected item, the adversary then has the power to remove any number of data items in the remaining set; this repeats until the remaining set is empty, which then terminates the algorithm.

For convenience, we often use a diagram to illustrate an adversarial strategy. A diagram of an adversarial strategy is an acyclic directed graph, where each node represents a possible state of the strategy, and each arc indicates a possible transition. Each state of the strategy contains two boxes. The first box indicates the current solution maintained by the algorithm, the second box indicates the remaining set of data items maintained by the adversary. A state can be either terminal or non-terminal. A state is *terminal* if and only if it is a sink, in the sense that the adversary no longer need perform any additional action; we indicate a terminal state using **bold** boxes. Each transition also contains two lines of actions. The first line indicates the action taken by the algorithm and the second line indicates the action taken by the adversary. Sometimes the algorithm may need to reject certain data items in order to accept a new one, so an action may contain multiple operations which occur at the same time; we use $\oslash$ if there is no action. Note that to calculate a bound for the approximation ratio of an algorithm, it is sufficient to consider the approximation ratios achieved in all terminal states. We will see such diagrams in Sect. 3.

3 Priority Algorithms and Approximation Bounds

We first define four constants that will be used for our results. Let α, β, γ and δ be the real roots (respectively) of the equations $2x^3+x^2-1=0$, $2x^2+x-2=0$, $10x^2-5x-3=0$ and $6x^2-2x-3=0$ between 0 and 1. The corresponding values are shown in Table 1.

Table 1. Corresponding values.

name	α	β	γ	δ
value	≈ 0.657	≈ 0.780	≈ 0.852	≈ 0.893

We now give a tight bound for irrevocable priority algorithms. It is interesting that there is no approximability difference between fixed order and adaptive irrevocable priority algorithms.

[6] Technically we can use an item number identifier to further distinguish items, but by providing sufficiently many copies of an item the adversary can effectively achieve what the statement claims.

Theorem 1. *There is a fixed order irrevocable priority algorithm for SSP with approximation ratio α, and every irrevocable priority algorithm for SSP has approximation ratio at most α.*

The case for revocable priority algorithms is more interesting. The ability to make revocable acceptances gives the algorithm a certain flexibility to "regret". The data items admitted into the solution are never "safe" until the termination of the algorithm. Therefore, if there is enough "room", it never hurts to accept a data item no matter how "bad" it is, as we can always reject it later at any time and with no cost. For the rest of the paper, we assume our algorithms will take advantage of this property. We start with fixed order revocable priority algorithm by giving two tight bounds for non-increasing order (i.e. items are ordered so that $u_1 \geq u_2 \ldots \geq u_n$) and non-decreasing order revocable priority algorithms.

Theorem 2. *There is a non-increasing order revocable priority algorithm for SSP that has approximation ratio α, and every such algorithm has approximation ratio at most α. (Note that the simple ordering here is different from the fixed order irrevocable algorithm of Theorem 1.)*

Theorem 3. *There is a non-decreasing order revocable priority algorithm for SSP that has approximation ratio β, and every such algorithm has approximation ratio at most β.*

The improvement using non-decreasing order is perhaps counter-intuitive [7] as one might think it is more flexible to fill in with small items at the end. Next, we give a approximation bound for any fixed order revocable priority algorithm; this exhibits the first approximation gap we are unable to close. The technique used in the proof is based on a chain of possible item priorities. It turns out, in order to achieve certain approximation ratio, some data items must be placed before some other data items. This order relation is transitive and therefore, has to be acyclic.

Theorem 4. *No fixed order revocable priority algorithm of SSP can achieve approximation ratio better than γ.*

Proof. Let $u_1 = 0.2$, $u_2 = \frac{1}{2}\gamma - \frac{1}{10} \approx 0.326$, $u_3 = 0.5$, and $u_4 = 0.8$. For a data item u, we denote by $rank(u)$ its priority. There are four cases:

1. If $rank(u_4) > rank(u_3)$, then the adversarial strategy is shown in Fig. 1. If the algorithm terminates via state $\mathbf{s}_1$, then
$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_3}{u_4} < \gamma.$$
If the algorithm terminates via state $\mathbf{s}_2$, then
$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_4}{2u_3} = u_4 < \gamma.$$
2. If $rank(u_3) > rank(u_2)$, then the adversarial strategy is shown in Fig. 2.

[7] As another example, in the identical machines makespan problem, it is provably advantageous to consider the largest items first.

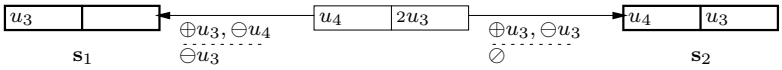

Fig. 1. Adversarial strategy for $rank(u_4) > rank(u_3)$.

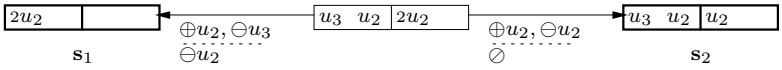

Fig. 2. Adversarial strategy for $rank(u_3) > rank(u_2)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{2u_2}{u_2 + u_3} = \frac{\gamma - \frac{1}{5}}{\frac{1}{2} + \frac{1}{2}\gamma - \frac{1}{10}} = \frac{10\gamma - 2}{5\gamma + 4} < \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_2 + u_3}{3u_2} = \frac{\frac{1}{2} + \frac{1}{2}\gamma - \frac{1}{10}}{\frac{3}{2}\gamma - \frac{3}{10}} = \frac{5\gamma + 4}{15\gamma - 3} < \gamma.$$

3. If $rank(u_2) > rank(u_1)$, then the adversarial strategy is shown in Fig. 3.

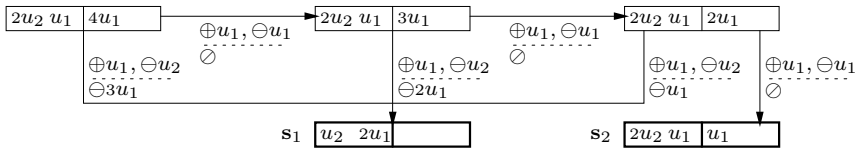

Fig. 3. Adversarial strategy for $rank(u_2) > rank(u_1)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{2u_1 + u_2}{u_1 + 2u_2} = \frac{\frac{2}{5} + \frac{1}{2}\gamma - \frac{1}{10}}{\frac{1}{5} + \gamma - \frac{1}{5}} = \frac{\frac{1}{2}\gamma + \frac{3}{10}}{\gamma} = \frac{5\gamma + 3}{10\gamma} = \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_1 + 2u_2}{5u_1} = u_1 + 2u_2 = \frac{1}{5} + \gamma - \frac{1}{5} = \gamma.$$

4. If $rank(u_1) > rank(u_2) > rank(u_3) > rank(u_4)$, then the adversarial strategy is shown in Fig. 4.

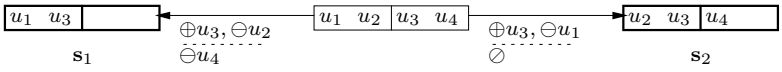

Fig. 4. Adversarial strategy for $rank(u_1) > rank(u_2) > rank(u_3) > rank(u_4)$.

If the algorithm terminates via state $\mathbf{s}_1$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{u_1 + u_3}{u_2 + u_3} = \frac{\frac{1}{5} + \frac{1}{2}}{\frac{1}{2}\gamma - \frac{1}{10} + \frac{1}{2}} = \frac{7}{5\gamma + 4} < \gamma.$$

If the algorithm terminates via state $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_2 + u_3}{u_1 + u_4} = u_2 + u_3 = \frac{1}{2}\gamma - \frac{1}{10} + \frac{1}{2} < \gamma.$$

As a conclusion, no fixed order revocable priority algorithm of SSP can achieve approximation ratio better than γ. This completes the proof. □

Finally, we study adaptive revocable priority algorithms. This is the strongest class of algorithms studied in this paper and arguably represents the ultimate approximation power of greedy algorithms (for packing problems). We show that no such algorithm can achieve an approximation ratio better than δ, and then we develop a relatively subtle algorithm having approximation ratio 0.8 in Theorem 6, thus leaving another gap in what is provably the best approximation ratio possible.

Theorem 5. *No adaptive revocable priority algorithm of SSP can achieve approximation ratio better than δ.*

Proof. Let $u_1 = \frac{1}{3}\delta \approx 0.298$ and $u_2 = 0.5$. For a given algorithm, we utilize the following adversary strategy shown in Fig. 5. If the algorithm terminates via state $\mathbf{s}_1$ or $\mathbf{s}_2$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{3u_1}{2u_2} = 3u_1 = \delta.$$

If the algorithm terminates via state $\mathbf{s}_3$ or $\mathbf{s}_4$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} \leq \frac{u_1 + u_2}{3u_1} = \frac{\frac{1}{3}\delta + \frac{1}{2}}{\delta} = \frac{2\delta + 3}{6\delta} = \delta.$$

If the algorithm terminates via state $\mathbf{s}_5$, then

$$\rho = \frac{\|\mathbf{ALG}\|}{\|\mathbf{OPT}\|} = \frac{2u_1}{u_1 + u_2} = \frac{\frac{2}{3}\delta}{\frac{1}{3}\delta + \frac{1}{2}} = \frac{4\delta}{2\delta + 3} < \delta.$$

In all three cases, the adversary forces the algorithm to have approximation ratio no better than δ; this completes the proof. □

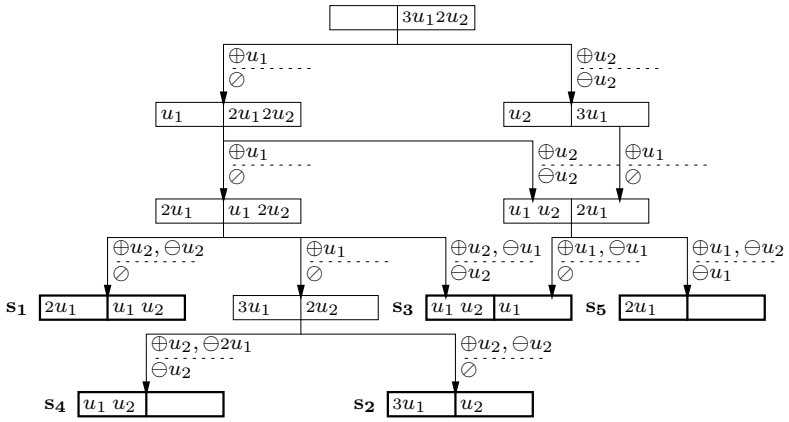

Fig. 5. Adversarial strategy for adaptive revocable priority algorithms.

Our 0.8 approximation adaptive priority algorithm is facilitated by a few simplifying observations. First of all, since we are targeting a ratio of 0.8, the algorithm can terminate whenever it detects an item u in the remaining input such that a subset of $(\mathbf{B} \oplus u)$ has total weight ≥ 0.8; such an item is *adaptively* given the highest priority and we call this a *terminal condition*. In our algorithm description, it is understood that the algorithm first *adaptively* checks for such a condition, and terminates if it is satisfied. Note that the running time of checking such condition is bounded by a constant. Secondly, data items outside $(0.2, 0.8)$ are not needed for the analysis of the algorithm. That is, items in $[0.8, 1]$ trivialize the problem and items in $(0, .2]$ can be considered at the end and added to $\mathbf{B}$ (if necessary) to achieve the desired bound. Finally, we assume the current set of accepted items $\mathbf{B}$ operates in one of the following four modes:

1. `Queue Mode:` In this mode, accepted items are discarded in the FIFO order to accommodate the new data item u.
2. `Queue_1 Mode:` In this mode, the first accepted item is never discarded, the rest data items are discarded in the FIFO order to accommodate the new data item u.
3. `Stack Mode:` In this mode, accepted items are discarded in the FILO order to accommodate the new data item u.
4. `Optimum Mode:` In this mode, accepted items are discarded to maximize $\|\mathbf{B}\|$; the new data item u may also be discarded for this purpose.

We use $\mathbf{B}_{\text{mode}}$ to represent the operational mode of $\mathbf{B}$. The algorithm can switch among these four modes during the processing of data items; we do not explicitly mention in the algorithm what data items are being discarded since it is well-defined under its operational mode.

The algorithm uses an ordering of data items which is determined by its distance to 0.3, i.e., the closer a data item to 0.3, the higher its priority is,

breaking tie arbitrarily. Note that by the first observation given earlier, this ordering may be interrupted if at any point of time a terminal condition is satisfied, so this is not a fixed order priority algorithm. The algorithm is described below.

Algorithm ADAPTIVE

B := $\emptyset$;
if the first data item is in $(0.2, 0.35)$ **then**
  $\mathbf{B}_{\text{mode}}$:= `Queue`;
else
  $\mathbf{B}_{\text{mode}}$:= `Queue_1`;
end if
while I contains a data item $\in (0.2, 0.4]$ **do**
  let u be the next data item in **I**;
  I := **I** $\ominus u$;
  accept u;
end while
if B contains exactly three data items and all are $\in (0.2, 0.3]$ **then**
  $\mathbf{B}_{\text{mode}}$:= `Stack`;
else
  $\mathbf{B}_{\text{mode}}$:= `Optimum`;
end if
while I contains a data item $\in (0.4, 0.8)$ **do**
  let u be the next data item in **I**;
  I := **I** $\ominus u$;
  accept u;
end while

Theorem 6. *Algorithm* ADAPTIVE *achieves approximation ratio 0.8 for SSP.*

It is seemingly a small step from a 0.78 algorithm to a 0.8 algorithm, but the latter algorithm requires a substantially more refined approach and detalied analysis. The merit, we believe, in studying such a class of "simple algorithms" is that the simplicity of the structure suggests algorithmic ideas and allows a careful analysis of such algorithms. Limiting ourselves to simple algorithmic forms and exploiting the flexibility within such forms may very well give us a better understanding of the structure of a given problem and a better chance to derive new algorithms.

4 Conclusion

We analyze different types of priority algorithms for SSP leaving open two approximability gaps, one for fixed order and one for adaptive revocable priority algorithms. It is interesting that such gaps and non-trivial algorithms exist for such a simple class of algorithms and such a basic problem as SSP. We optimistically believe that surprisingly good algorithms can be designed within the

revocable priority framework for problems which are currently not well understood.

References

1. Angelopoulos, S. and Borodin, A.: On the power of priority algorithms for facility location and set cover. Algorithmica **40** (2004) 271–291
2. Arrow, K.: Social Choice and Individual Values. Wiley (1951)
3. Baptiste, P.: Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine with equal processing times. Journal of Scheduling **2** (1999) 245–252
4. Bar-Noy, A., Guha, S., Naor, J. and Schieber, B.: Approximating throughput in real-time scheduling. SIAM Journal of Computing **31** (2001) 331–352
5. Borodin, A., Nielsen, M., and Rackoff C.: (Incremental) priority algorithms. Algorithmica **37** (2003) 295–326
6. Borodin, A., Boyar, J., and Larsen K.: Priority algorithms for graph optimization problems. Lecture Notes in Computer Science **3351** (2005) 126–139
7. Chrobak, M., Durr, C., Jawor, W., Kowalik, L. and Kurowski, M. On scheduling equal length jobs to maximize throughput. To appear in Journal of Scheduling.
8. Chuzhoy, J., Ostrovsky, R. and Rabani, Y.: Approximation algorithms for the job interval scheduling problem and related scheduling problems. In Proceedings of 42nd Annual IEEE Symposium of Foundations of Computer Science (2001) 348–356
9. Davis, S. and Impagliazzo, R.: Models of greedy algorithms for graph problems. In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (2004) 381–390
10. Erlebach, T. and Spieksma, F.: Interval selection: Applications, algorithms, and lower bounds. Journal of Algorithms **46** (2003) 27–53
11. Garey, M. and Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman (1979)
12. Horn, S.: One-pass algorithms with revocable acceptances for job interval selection. Master's thesis, University of Toronto (2004)
13. Ibarra, O. and Kim, C.: Fast approximation algorithms for the knapsack and sum of subset problem. Journal of the ACM **22** (1975) 463–468
14. Iwama, K. and Taketomi, S.: Removable online knapsack problems. Lecture Notes in Computer Science **2380** (2002) 293–305
15. Kellerer, H., Mansini, R., Pferschy, U. and Speranza, M.: An efficient fully polynomial approximation scheme for the subset-sum problem. Journal of Computer and System Science **66** (2003) 349–370
16. Martello, S. and Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley and Sons (1990)
17. Moore, J.: An n-job, one machine sequencing algorithm algorithm for minimizing the number of late jobs. Management Science **15** (1968) 102–109
18. Regev, O.: Priority algorithms for makespan minimization in the subset model. Information Processing Letters **84** (2002) 153–157
19. Ye, Y. and Borodin, A.: Priority algorithms for the subset-sum problem. Technical Report, University of Toronto (2007) *http://www.cs.toronto.edu/~bor*