On the approximation ability of evolutionary optimization with application to minimum set cover

Yang Yu [a], Xin Yao [b], Zhi-Hua Zhou [a,*]

[a] *National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*
[b] *Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK*

ARTICLE INFO

ABSTRACT

Evolutionary algorithms (EAs) are heuristic algorithms inspired by natural evolution. They are often used to obtain satisficing solutions in practice. In this paper, we investigate a largely underexplored issue: the approximation performance of EAs in terms of how close the solution obtained is to an optimal solution. We study an EA framework named simple EA with isolated population (SEIP) that can be implemented as a single- or multi-objective EA. We analyze the approximation performance of SEIP using the partial ratio, which characterizes the approximation ratio that can be guaranteed. Specifically, we analyze SEIP using a set cover problem that is NP-hard. We find that in a simple configuration, SEIP efficiently achieves an H_n-approximation ratio, the asymptotic lower bound, for the unbounded set cover problem. We also find that SEIP efficiently achieves an $(H_k - \frac{k-1}{8k^9})$-approximation ratio, the currently best-achievable result, for the k-set cover problem. Moreover, for an instance class of the k-set cover problem, we disclose how SEIP, using either one-bit or bit-wise mutation, can overcome the difficulty that limits the greedy algorithm.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Evolutionary algorithms (EAs) [3] have been successfully applied to many fields and can achieve extraordinary performance in addressing some real-world hard problems, particularly NP-hard problems [16,18,17,4]. To gain an understanding of the behavior of EAs, many theoretical studies have focused on the running time required to achieve exact optimal solutions [14,33,26,2]. In practice, EAs are most commonly used to obtain *satisficing* solutions, yet theoretical studies of the approximation ability of EAs have only emerged recently.

He and Yao [15] first studied conditions under which the *wide-gap far distance* and the *narrow-gap long distance* problems are hard to approximate using EAs. Giel and Wegener [12] investigated a $(1+1)$-EA for a *maximum matching* problem and found that the time taken time to find exact optimal solutions is exponential but is only $O(n^{2\lceil 1/\epsilon \rceil})$ for $(1+\epsilon)$-approximate solutions, which demonstrates the value of EAs as approximation algorithms.

Subsequently, further results on the approximation ability of EAs were reported. For the $(1+1)$-EA, the simplest type of EA, two classes of results have been obtained. On one hand, it was found that the $(1+1)$-EA has an arbitrarily poor approximation ratio for the *minimum vertex cover* problem and thus also for the *minimum set cover* problem [11,28]. On the other hand, it was also found that $(1+1)$-EA provides a *polynomial-time randomized approximation scheme* for a subclass of the *partition* problem [32]. Furthermore, for some subclasses of the minimum vertex cover problem for which the $(1+1)$-EA gets stuck, a multiple restart strategy allows the EA to recover an optimal solution with high probability [28]. Another result

---

* Corresponding author.
   E-mail addresses: yuy@lamda.nju.edu.cn (Y. Yu), x.yao@cs.bham.ac.uk (X. Yao), zhouzh@nju.edu.cn (Z.-H. Zhou).

for the $(1 + 1)$-EA is that it improves a 2-approximation algorithm to a $(2 - 2/n)$-approximation on the minimum vertex cover problem [10]. This implies that it might sometimes be useful as a post-optimizer.

Recent advances in multi-objective (usually bi-objective) EAs have shed light on the power of EAs as approximation optimizers. For a single-objective problem, *multi-objective reformulation* introduces an auxiliary objective function for which a multi-objective EA is used as the optimizer. Scharnow et al. [30] first suggested that multi-objective reformulation could be superior to use of a single-objective EA. This was confirmed for various problems [24,25,11,27] by showing that while a single-objective EA could get stuck, multi-objective reformulation helps to solve the problems efficiently.

Regarding approximations, it has been shown that multi-objective EAs are effective for some NP-hard problems. Friedrich et al. [11] proved that a multi-objective EA achieves a $(\ln n)$-approximation ratio for the minimum set cover problem, and reaches the asymptotic lower bound in polynomial time. Neumann and Reichel [23] showed that multi-objective EAs achieve a k-approximation ratio for the *minimum multicuts* problem in polynomial time.

In the present study, we investigate the approximation ability of EAs by introducing a framework called *simple evolutionary algorithm with isolated population* (SEIP), which uses an isolation function to manage competition among solutions. By specifying the isolation function, SEIP can be implemented as a single- or multi-objective EA. Multi-objective EAs previously analyzed [22,11,23] can be viewed as special cases of SEIP in term of the solutions maintained in the population. By analyzing the SEIP framework, we obtain a general characterization of EAs that guarantee approximation quality.

We then study the minimum set cover problem (MSCP), which is an NP-hard problem [9]. We prove that for the unbounded MSCP, a simple configuration of SEIP efficiently obtains an H_k-approximation ratio (where H_k is the harmonic number of the cardinality of the largest set), the asymptotic lower bound [9]. For the minimum k-set cover problem, this approach efficiently yields an $(H_k - \frac{k-1}{8k^9})$-approximation ratio, the currently best-achievable quality [13]. Moreover, for a subclass of the minimum k-set cover problem, we demonstrate how SEIP, with either one-bit or bit-wise mutation, can overcome the difficulty that limits the greedy algorithm.

The remainder of the paper is organized as follows. After introducing preliminaries in Section 2, we describe SEIP in Section 3 and characterize its behavior for approximation in Section 4. We then analyze the approximation ratio achieved by SEIP for the MSCP in Section 5. In Section 6 we conclude the paper with a discussion of the advantages and limitations of the SEIP framework.

2. Preliminaries

We use bold small letters such as $\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$ to represent vectors. We denote $[m]$ as the set $\{1, 2, \ldots, m\}$ and 2^S as the power set of S, which consists of all subsets of S. We denote $H_n = \sum_{i=1}^{n} \frac{1}{i}$ for the nth harmonic number. Note that $H_n \sim \ln n$ since $\ln n \leqslant H_n \leqslant \ln n + 1$.

In this paper, we consider minimization problems as follows.

Definition 1 *(Minimization problem).* Given an evaluation function f and a set of feasibility constraints $\mathcal{C}$, find a solution $\boldsymbol{x} \in \{0, 1\}^n$ that minimizes $f(\boldsymbol{x})$ while satisfying constraints in $\mathcal{C}$. A problem instance can be specified by its parameters $(n, f, \mathcal{C})$.

In the definition of the minimization problem, solutions are represented in binary vectors. When the aim of a minimization problem is to find a subset from a universal set, we can equivalently use a binary vector to represent a subset, where each element of the vector indicates the membership of a corresponding element of the universe set. For example, given a universal set $U = \{1, 2, 3, 4, 5\}$, its subset $S = \{1, 3, 5\}$ can be represented by a binary vector $\boldsymbol{v} = (1, 0, 1, 0, 1)$, and we define $U(\boldsymbol{v}) = S$. Considering the equivalence between sets and binary vectors, we apply set operators $(|\cdot|, \cap, \cup, -, \subseteq, \in)$ to binary vectors when there is no confusion. For example, $|(1, 0, 1, 0, 1)| = 3$, $(1, 0, 1, 0, 1) \cap (0, 0, 0, 1, 1) = (0, 0, 0, 0, 1)$, and $(1, 0, 1, 0, 1) - (0, 0, 0, 1, 1) = (1, 0, 1, 0, 0)$. We denote $\boldsymbol{x}^\emptyset = (0, \ldots, 0)$ as the vector corresponding to the empty set.

We investigate the minimum set cover problem (MSCP), which is an NP-hard problem.

Definition 2 *(Minimum set cover problem (MSCP)).* Given a set of n elements $U = [n]$ and a collection $C = \{S_1, S_2, \ldots, S_m\}$ of m nonempty subsets of U, where each S_i is associated with a positive cost $w(S_i)$, find a subset $X^* \subseteq C$ such that $\sum_{S \in X^*} w(S)$ is minimized with respect to $\bigcup_{S \in X^*} S = U$.

Using binary vector representation, we denote an instance of the MSCP by its parameters $(n, \boldsymbol{w}, C, U)$, where $|C| = m$ and $\boldsymbol{w}$ is the cost vector. The MSCP involves finding a vector $\boldsymbol{x}^{OPT}$, which is equivalent to its set representation X^*, by solving a constrained optimization problem

$$\boldsymbol{x}^{OPT} = \arg\min_{\boldsymbol{x} \in \{0, 1\}^m} \boldsymbol{w} \cdot \boldsymbol{x}$$

$$\text{s.t.} \bigcup_{S \in C(\boldsymbol{x})} S = U,$$

where $\boldsymbol{w} \cdot \boldsymbol{x}$ is the inner product between vectors $\boldsymbol{w}$ and $\boldsymbol{x}$, and $C(\boldsymbol{x})$ denotes a set consisting of elements in the collection C that are indicated by binary vector $\boldsymbol{x}$.

Definition 3 *(Minimum k-set cover problem).* An MSCP $(n, \boldsymbol{w}, C, U)$ is a k-set cover problem if, for some constant k, it holds that $|S| \leqslant k$ for all $S \in C$, denoted as $(n, \boldsymbol{w}, C, U, k)$.

A solution is called *feasible* if it satisfies all the constraints in C; otherwise, it is called an *infeasible* solution, or a *partial* solution in this paper. Here, we assume that the evaluation function f is defined on the full input space, that is, it evaluates all possible solutions regardless of their feasibility. For example, if we evaluate a solution $\boldsymbol{x}$ of the MSCP using $\boldsymbol{w} \cdot \boldsymbol{x}$, this evaluation function can be calculated for any solution.

Given a minimization problem, we denote $\boldsymbol{x}^{OPT}$ as an optimal solution of the problem, and $OPT = f(\boldsymbol{x}^{OPT})$. For a feasible solution $\boldsymbol{x}$, we regard the ratio $\frac{f(\boldsymbol{x})}{OPT}$ as its *approximation ratio*. If the approximation ratio of a feasible solution is upper-bounded by some value r, that is,

$$1 \leqslant \frac{f(\boldsymbol{x})}{OPT} \leqslant r,$$

the solution is called an *r-approximate solution*. An algorithm that guarantees to find an r-approximate solution for an arbitrary problem instance in polynomial time is an *r-approximation algorithm*.

The greedy algorithm [5] described here as Algorithm 1 is the most well-known approximation algorithm for the MSCP. This greedy algorithm consists of a sequence of steps. The cost of a candidate set is defined as its weight divided by the number of its elements that have not been covered yet (i.e., the quantity r_S in line 3). The algorithm then picks the candidate set with the smallest cost for the solution (line 4) and marks the newly covered elements (line 6). This simple algorithm yields an H_n-approximation ratio or, more exactly, H_k, where k is the cardinality of the largest set [5]. The key to the proof of the approximation ratio is the definition of the *price* of elements, as in line 5. The *price* of an element equals the cost of the set that first covers it; therefore, the total *price* of all elements equals the total cost of the solution. Furthermore, it should be noted that when an element is covered by a set with the lowest cost, it would also be covered by one set of an optimal solution but with a higher cost. Therefore the *price* of the element is upper-bounded by the optimal cost and hence the approximation ratio is upper-bounded. For a detailed proof please refer to Cheátal [5].

Algorithm 1 *(Greedy algorithm).* (See [5].)
Given a minimum k-set cover problem $(n, \boldsymbol{w}, C, U, k)$, the greedy algorithm consists of the following steps:

1: $X \leftarrow \emptyset; R \leftarrow \emptyset$
2: **while** $R \neq U$ **do**
3:    $\forall S \in C: |S - R| > 0$, let $r_S \leftarrow \frac{w(S)}{|S-R|}$
4:    $\hat{S} \leftarrow \arg\min_S r_S$
5:    let $price(e) \leftarrow r_{\hat{S}}$ for all $e \in \hat{S} - R$
6:    let $R \leftarrow R \cup \hat{S}$, and $X \leftarrow X \cup \{\hat{S}\}$.
7: **end while**
8: **return** X

Several studies have shown that the approximation ratio of the MSCP is lower-bounded by $\Omega(\ln n)$ unless $P = NP$ that is unlikely [29,31,9,1]. Therefore, the greedy algorithm achieves the asymptotic lower bound of the approximation ratio for the MSCP.

Although the H_n-approximation ratio is asymptotically tight for the unbounded MSCP, a better approximation ratio can be achieved for the minimum k-set cover problem, where k is a constant. It has been proved that for the unweighted minimum k-set cover problem, an $(H_k - \frac{1}{2})$-approximation ratio can be achieved [8] and, if $k \geqslant 4$, an improved ratio $(H_k - \frac{196}{390})$ can be achieved [20]. For the weighted minimum k-set cover problem, a greedy-algorithm-with-withdrawals (GAWW) was presented and achieved an $H_k - \frac{k-1}{8k^9}$-approximation ratio [13].

The GAWW algorithm presented here as Algorithm 2 is a modification of the greedy algorithm. In every iteration, the algorithm chooses between taking a greedy step, which is the same as in the greedy algorithm, and a withdrawal step, which replaces a set in the current solution with at most k candidate sets. It evaluates the cost of candidate sets as in the greedy algorithm, and also evaluates the benefit of the withdrawal (calculated in lines 4 and 5). When the benefit of the withdrawal step is not large enough according to the criterion in line 6, the algorithm takes the greedy step, and otherwise takes the withdrawal step. To prove the approximation ratio, the *price* of elements is defined similarly in line 7 for the greedy step and line 10 for the withdrawal step, which is used later in the proofs for this paper.

Algorithm 2 *(GAWW).* (See [13].)
Given a minimum k-set cover problem $(n, \boldsymbol{w}, C, U, k)$, GAWW consists of the following steps.

1: $X \leftarrow \emptyset; R \leftarrow \emptyset; \alpha_k \leftarrow 1 - \frac{1}{k^3}$

2: **while** $R \neq U$ **do**
3:     $\forall S \in C : |S - R| > 0$, let $r_S \leftarrow \frac{w(S)}{|S-R|}$
4:     $\forall S \in X, Q \subseteq C : |Q| \leqslant k \wedge |\bigcup_{S' \in Q} S' - R| > 0$, let $r_{(S,Q)} \leftarrow \frac{\sum_{S' \in Q} c_{S'} - c_S}{|\bigcup_{S' \in Q} S' - R|}$
5:     $\hat{S} \leftarrow \arg\min_S r_{\hat{S}}$, and $(\tilde{S}, \tilde{Q}) \leftarrow \arg\min_{(S,Q):|Q| \leqslant k} r_{(S,Q)}$ {choose the minimal number of sets in cases of ties}
6:     **if** $r_{\hat{S}} \cdot \alpha_k \leqslant r_{(\tilde{S}, \tilde{Q})}$ **then** {greedy step}
7:        let $price(e) \leftarrow r_{\hat{S}}$ for all $e \in \hat{S} - R$
8:        let $R \leftarrow R \cup \hat{S}$, and $X \leftarrow X \cup \{\hat{S}\}$
9:     **else** {i.e., $r_{(\tilde{S}, \tilde{Q})} < r_{\hat{S}} \cdot \alpha_k$, withdrawal step}
10:        let $price(e) \leftarrow r_{(\tilde{S}, \tilde{Q})}$ for all $e \in \bigcup_{S \in Q} S - R$
11:        let $R \leftarrow \bigcup_{S \in Q} S \cup R$, and $X \leftarrow X \cup Q - \{\tilde{S}\}$
12:     **end if**
13: **end while**
14: **return** X

The $(1 + 1)$-EA is the simplest EA implementation, as described in Algorithm 3. Starting from a solution generated uniformly at random, the $(1 + 1)$-EA repeatedly generates a new solution from the current one using a mutation operator, and the current solution is replaced if the new solution has better (or equal) fitness.

Algorithm 3 *((1 + 1)-EA).*
Given a minimization problem $(n, f, \mathcal{C})$, each solution is encoded as a binary vector of length m and the $(1 + 1)$-EA-minimizing f consists of the following steps.
1: $\boldsymbol{x} \leftarrow$ a solution generated uniformly at random
2: **while** stop $\neq$ true **do**
3:     Mutate $\boldsymbol{x}$ to generate $\boldsymbol{x}'$
4:     **if** x is feasible and $f(\boldsymbol{x}') \leqslant f(\boldsymbol{x})$ **then**
5:        $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
6:     **end if**
7: **end while**
8: **return** $\boldsymbol{x}$

Two mutation operators are commonly used to implement the "mutate" step in line 3:

One-bit mutation: Flip one randomly selected bit position of $\boldsymbol{x}$ to generate $\boldsymbol{x}'$.
Bit-wise mutation: Flip each bit of $\boldsymbol{x}$ with probability $\frac{1}{m}$ to generate $\boldsymbol{x}'$.

It has been shown that the $(1+1)$-EA has an arbitrarily poor approximation ratio for the MSCP [11]. Laumanns et al. [19] used a multi-objective reformulation with a multi-objective EA named SEMO to achieve a $(\ln n)$-approximation ratio. The SEMO algorithm is described in Algorithm 4, where two objectives are presented as (f_1, f_2). To apply SEMO to the MSCP, let f_1 evaluate the cost of the solution and f_2 evaluate the number of uncovered elements. Thus, SEMO minimizes the cost and the number of uncovered elements of solutions simultaneously. A notable difference between SEMO and $(1 + 1)$-EA is that SEMO uses a non-dominance relationship, implemented using the dominate function in SEMO. The population of SEMO maintains non-dominant solutions, that is, no solution is superior to another for both of the two objectives.

Algorithm 4 *(SEMO).* (See [19].)
Given a two-objective minimization problem $(n, (f_1, f_2))$, each solution is encoded as a binary vector of length m. SEMO minimization of (f_1, f_2) consists of the following steps.
1: $P \leftarrow \{\boldsymbol{x}^{\emptyset} = (0, 0, \ldots, 0)\}$
2: **while** stop $\neq$ true **do**
3:     Choose $\boldsymbol{x} \in P$ uniformly at random
4:     Mutate $\boldsymbol{x}$ to generate $\boldsymbol{x}'$
5:     **if** $\forall \boldsymbol{x} \in P$: dominate$(\boldsymbol{x}, \boldsymbol{x}') =$ false **then**
6:        $Q \leftarrow \{\boldsymbol{x} \in P \mid$ dominate$(\boldsymbol{x}', \boldsymbol{x}) =$ true$\}$
7:        $P \leftarrow P \cup \{\boldsymbol{x}'\} - Q$
8:     **end if**
9: **end while**
10: **return** P
where the dominate function of two solutions is defined such that dominate$(\boldsymbol{x}, \boldsymbol{y})$ is true if any one of the following three rules is satisfied:

1) $f_1(\boldsymbol{x}) < f_1(\boldsymbol{y})$ and $f_2(\boldsymbol{x}) \leqslant f_2(\boldsymbol{y})$
2) $f_1(\boldsymbol{x}) \leqslant f_1(\boldsymbol{y})$ and $f_2(\boldsymbol{x}) < f_2(\boldsymbol{y})$
3) $f_1(\boldsymbol{x}) = f_1(\boldsymbol{y})$ and $f_2(\boldsymbol{x}) = f_2(\boldsymbol{y})$ and $|\boldsymbol{x}| < |\boldsymbol{y}|$
and dominate($\boldsymbol{x}, \boldsymbol{y}$) is false otherwise.

3. SEIP

The SEIP framework is depicted in Algorithm 5. It uses an *isolation function* μ to isolate solutions. For some integer q, the function μ maps a solution to a subset of $[q]$. If and only if two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are mapped to subsets with the same cardinality, that is, $|\mu(\boldsymbol{x}_1)| = |\mu(\boldsymbol{x}_2)|$, the two solutions compete with each other. In that case, we say the two solutions are in the same *isolation*, and there are at most $q + 1$ isolations since the subsets of $[q]$ have $q + 1$ different cardinalities.

Algorithm 5 *(SEIP).*
Given a minimization problem $(n, f, \mathcal{C})$, an isolation function μ encodes each solution as a binary vector of length m. SEIP minimization of f with respect to constraint $\mathcal{C}$ consists of the following steps.

1: $P \leftarrow \{\boldsymbol{x}^{\emptyset} = (0, 0, \ldots, 0)\}$
2: **while** stop $\neq$ true **do**
3:    Choose $\boldsymbol{x} \in P$ uniformly at random
4:    Mutate $\boldsymbol{x}$ to generate $\boldsymbol{x}'$
5:    **if** $\forall \boldsymbol{x} \in P$: superior($\boldsymbol{x}, \boldsymbol{x}'$) = false **then**
6:       $Q \leftarrow \{\boldsymbol{x} \in P \mid$ superior($\boldsymbol{x}', \boldsymbol{x}$) = true$\}$
7:       $P \leftarrow P \cup \{\boldsymbol{x}'\} - Q$
8:    **end if**
9: **end while**
10: **return** the best feasible solution in P

where the superior function of two solutions determines whether one solution is superior to the other. This is defined as follows: superior($\boldsymbol{x}, \boldsymbol{y}$) is true if both of the following rules are satisfied:
1) $|\mu(\boldsymbol{x})| = |\mu(\boldsymbol{y})|$
2) $f(\boldsymbol{x}) < f(\boldsymbol{y})$, or $f(\boldsymbol{x}) = f(\boldsymbol{y})$ but $|\boldsymbol{x}| < |\boldsymbol{y}|$
and superior($\boldsymbol{x}, \boldsymbol{y}$) is false otherwise.

When the isolation function puts all solutions in an isolation for a particular instance, it degrades to the $(1 + 1)$-EA. The isolation function can also be configured to simulate the dominance relationship of multi-objective EAs such as SEMO/GSEMO [19] and DEMO [23]. If we are dealing with k-objective optimization with discrete objective values, a simple approach is to use one of the objective functions, say f_1, as the fitness function and use the combination of the values of the remaining $k - 1$ objective functions (say $f_2, \ldots, f_k$) as the isolation functions. Thus, two solutions compete (for f_1) only when they share the same objective values for $f_2, \ldots, f_k$. This simulation shows that all the non-dominant solutions of a multi-objective EA are also kept in the SEIP population, and if a non-dominant solution does not reside in the SEIP population, there must be another solution that dominates it. Hence, SEIP can be viewed as a generalization of multi-objective EAs in terms of the solutions retained. This simulation also reveals that SEIP retains more solutions than a multi-objective EA using the dominance relationship. This, on one hand, SEIP takes more time to manipulate a larger population than a multi-objective EA does, which could be overcome by defining an isolation function that aggregates nearby solutions, as has been done for DEMO [23]. On the other hand, SEIP has more opportunities available to find a better approximation solution, since the relationship "$\boldsymbol{a}$ dominates $\boldsymbol{b}$" does not imply that $\boldsymbol{a}$ definitely leads to a better approximation solution than $\boldsymbol{b}$.

Taking the MSCP $(n, \boldsymbol{w}, C, U)$ as an example, we can use the fitness function $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$, which is the sum of costs of the selected sets. For the isolation function, we can use $\mu(\boldsymbol{x})$ as $\emptyset$ if $\boldsymbol{x}$ is feasible and $\{1\}$ if $\boldsymbol{x}$ is infeasible, which isolates the feasible from the infeasible solutions (and thus $q = 1$); we can also use the isolation function $\mu(\boldsymbol{x}) = \bigcup_{S \in \boldsymbol{x}(C)} S$, and thus the solutions compete only when they cover the same number of elements (and thus $q = n$).

The mutation operator can use either one-bit or bit-wise mutation. Usually, one-bit mutation is considered for local searches, while bit-wise mutation is suitable for global searches as it has a positive probability for producing any solution. We denote SEIP with one-bit mutation as LSEIP and SEIP with bit-wise mutation as GSEIP, where "L" and "G" denote "local" and "global", respectively.

For convenience, SEIP is set to start from solution $\boldsymbol{x}^{\emptyset}$ rather than from a random solution, as commonly done for other EAs. Under the condition that any solution will have a better fitness if any 1 bit is turned to 0, we can bound the difference between random initialization and starting from $\boldsymbol{x}^{\emptyset}$. From a random solution, SEIP takes at most $O(qm \ln m)$ expected steps to find $\boldsymbol{x}^{\emptyset}$ according to the following argument. Suppose the worst case whereby random initialization generates a solution with all 1 bits; according to the fitness function condition, finding $\boldsymbol{x}^{\emptyset}$ is equivalent to solving the OneMax problem using a randomized local search and the $(1 + 1)$-EA, which takes $O(m \ln m)$ steps for both LSEIP and GSEIP [7]. Furthermore, note that there can be at most q solutions in the population, and thus it costs SEIP q expected steps to choose one particular solution from the population.

The stop criterion is not described in the definition of SEIP, since EAs are usually used as anytime algorithms in practice. We now analyze the approximation ratio and the corresponding computational complexity of SEIP.

4. General approximation behavior of SEIP

For minimization problems, we consider *linearly additive* isolation functions. μ is a linearly additive isolation function if, for some integer q,

$$\begin{cases} \mu(\boldsymbol{x}) = [q], & \text{for all feasible solutions } \boldsymbol{x}, \\ \mu(\boldsymbol{x} \cup \boldsymbol{y}) = \mu(\boldsymbol{x}) \cup \mu(\boldsymbol{y}), & \text{for all solutions } \boldsymbol{x} \text{ and } \boldsymbol{y}. \end{cases}$$

The quality of a feasible solution is measured in terms of the approximation ratio. To measure the quality of a partial (infeasible) solution, we define a *partial reference function* and *partial ratio* as follows.

Definition 4 *(Partial reference function).* Given a set $[q]$ and a value v, a function $\mathcal{L}_{[q],v} : 2^{[q]} \to \mathbb{R}$ is a partial reference function if

1) $\mathcal{L}_{[q],v}([q]) = v$,
2) $\mathcal{L}_{[q],v}(R_1) = \mathcal{L}_{[q],v}(R_2)$ for all $R_1, R_2 \subseteq [q]$ such that $|R_1| = |R_2|$.

For a minimization problem with optimal cost *OPT* and an isolation function μ mapping feasible solutions to the set $[q]$, we denote a partial reference function with respect to $[q]$ and *OPT* as $\mathcal{L}_{[q],OPT}$. When the problem and the isolation function are clear, we omit the subscripts and simply denote the partial reference function as $\mathcal{L}$.

Definition 5 *(Partial ratio).* Given a minimization problem $(n, f, \mathcal{C})$ and an isolation function μ, the *partial ratio* of a (partial) solution $\boldsymbol{x}$ with respect to a corresponding partial reference function $\mathcal{L}$ is

$$\text{p-ratio}(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{x}))},$$

and the *conditional partial ratio* of $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$ is

$$\text{p-ratio}(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{f(\boldsymbol{y} \mid \boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{y}) \mid \mu(\boldsymbol{x}))},$$

where $f(\boldsymbol{y} \mid \boldsymbol{x}) = f(\boldsymbol{x} \cup \boldsymbol{y}) - f(\boldsymbol{x})$ and $\mathcal{L}(\mu(\boldsymbol{y}) \mid \mu(\boldsymbol{x})) = \mathcal{L}(\mu(\boldsymbol{y}) \cup \mu(\boldsymbol{x})) - \mathcal{L}(\mu(\boldsymbol{x}))$.

The partial ratio is an extension of the approximation ratio. Note that the partial ratio for a feasible solution equals its approximation ratio. We have two properties of the partial ratio. One is that it is non-increasing in SEIP, as stated in Lemma 1, and the other is its decomposability, as stated in Lemma 2.

Lemma 1. *Given a minimization problem $(n, f, \mathcal{C})$ and an isolation function μ, if SEIP has generated an offspring $\boldsymbol{x}$ with partial ratio p with respect to a corresponding partial reference function $\mathcal{L}$, then there is a solution $\boldsymbol{y}$ in the population such that $|\mu(\boldsymbol{y})| = |\mu(\boldsymbol{x})|$, and the partial ratio of $\boldsymbol{y}$ is at most p.*

Proof. $\boldsymbol{x}$ is put into the population after it is generated; otherwise there is another solution $\boldsymbol{x}'$ with $|\mu(\boldsymbol{x}')| = |\mu(\boldsymbol{x})|$ and $f(\boldsymbol{x}') \leqslant f(\boldsymbol{x})$, and in this case let $\boldsymbol{x} = \boldsymbol{x}'$. The lemma is proved since $\mathcal{L}(\boldsymbol{x}) = \mathcal{L}(\boldsymbol{y})$ and by the `superior` function the cost is non-increasing. $\square$

From Lemma 1, we know that the partial ratio in each isolation remains non-increasing. Since SEIP repeatedly tries to generate solutions in each isolation, SEIP can be considered as optimizing the partial ratio in each isolation.

Lemma 2. *Given a minimization problem $(n, f, \mathcal{C})$ and an isolation function μ, for three (partial) solutions $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ such that $\boldsymbol{z} = \boldsymbol{x} \cup \boldsymbol{y}$, we have*

$$\text{p-ratio}(\boldsymbol{z}) \leqslant \max\{\text{p-ratio}(\boldsymbol{x}), \text{p-ratio}(\boldsymbol{y} \mid \boldsymbol{x})\},$$

with respect to a corresponding partial reference function $\mathcal{L}$.

Proof. Since $\boldsymbol{z} = \boldsymbol{x} \cup \boldsymbol{y}$, we have, by definition,

$$f(\boldsymbol{z}) = f(\boldsymbol{x}) + f(\boldsymbol{y} \mid \boldsymbol{x}),$$

and

$$\mu(\boldsymbol{z}) = \mu(\boldsymbol{x} \cup \boldsymbol{y}) = \mu(\boldsymbol{x}) \cup \mu(\boldsymbol{y}).$$

Thus, we have

$$\text{p-ratio}(\boldsymbol{z}) = \frac{f(\boldsymbol{z})}{\mathcal{L}(\mu(\boldsymbol{z}))} = \frac{f(\boldsymbol{x}) + f(\boldsymbol{y} \mid \boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{x}) \cup \mu(\boldsymbol{y}))} = \frac{f(\boldsymbol{x}) + f(\boldsymbol{y} \mid \boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{x})) + \mathcal{L}(\mu(\boldsymbol{y}) \mid \mu(\boldsymbol{x}))}$$

$$\leqslant \max\left\{ \frac{f(\boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{x}))}, \frac{f(\boldsymbol{y} \mid \boldsymbol{x})}{\mathcal{L}(\mu(\boldsymbol{y}) \mid \mu(\boldsymbol{x}))} \right\}$$

$$= \max\{\text{p-ratio}(\boldsymbol{x}), \text{p-ratio}(\boldsymbol{y} \mid \boldsymbol{x})\}. \quad \square$$

Lemma 2 reveals that the partial ratio for a solution is related to the conditional partial ratio of a building block. This can be considered as the way in which SEIP optimizes the partial ratio in each isolation, that is, by optimizing the conditional partial ratio of each building block partial solution. We then have the following theorem.

Theorem 1. *Given a minimization problem* $(n, f, \mathcal{C})$ *and an isolation function* μ *mapping to subsets of* $[q]$, *assume that every solution is encoded in an m-length binary vector. For some constant* $r \geqslant 1$ *with respect to a corresponding partial reference function* $\mathcal{L}$, *if*

1. $\text{p-ratio}(\boldsymbol{x}^{\emptyset}) \leqslant r$,
2. *for every partial solution* $\boldsymbol{x}$ *such that* $\text{p-ratio}(\boldsymbol{x}) \leqslant r$, *SEIP takes* $\boldsymbol{x}$ *as the parent solution and generates an offspring partial solution* $\boldsymbol{y}$ *such that* $\mu(\boldsymbol{x}) \subset \mu(\boldsymbol{x} \cup \boldsymbol{y})$ *and* $\text{p-ratio}(\boldsymbol{y} \mid \boldsymbol{x}) \leqslant r$ *in polynomial time in q and m*,

then SEIP finds an r-approximate solution in polynomial time in q and m.

Proof. Starting from $\boldsymbol{x}^{\emptyset}$, we can find a sequence of partial solutions $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{\ell}$, such that

$$\boldsymbol{x} = \bigcup_{i=1}^{\ell} \boldsymbol{x}_i \text{ is feasible,}$$

and that

$$\forall i = 1 \dots \ell: \text{p-ratio}\left(\boldsymbol{x}_i \mid \boldsymbol{x}^{\emptyset} \bigcup_{j=1}^{i-1} \boldsymbol{x}_j \right) \leqslant r,$$

because of the conditions. Note that when a partial solution is added to the solution, the offspring solution is in a different isolation to the parent solution. Since the isolation function is linearly additive, the length of the sequence ℓ cannot be greater than the number of isolations $q + 1$.

Let t be the time expected for SEIP to generate a partial solution $\boldsymbol{x}_i$ in the sequence from its parent solution, which is polynomial to q and m by the condition. It takes at most $O(q)$ expected steps for SEIP to pick the parent, since there are at most $q + 1$ solutions in the population. Therefore, the total time to reach a feasible solution $\boldsymbol{x}$ is $O(t \cdot q \cdot \ell)$, that is, $O(t \cdot q^2)$, which is still polynomial in q and m.

By Lemma 2, since the feasible solution $\boldsymbol{x}$ is composed of $\boldsymbol{x}^{\emptyset}$ and partial solutions $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_m$, the approximation ratio for $\boldsymbol{x}$ is at most as large as the maximum conditional partial ratio for the partial solutions, r. $\quad \square$

Theorem 1 reveals how SEIP can work to achieve an approximate solution. Starting from the empty set, SEIP uses its mutation operator to generate solutions in all isolations, and finally generates a feasible solution. During the process, SEIP repeatedly optimizes the partial ratio in each isolation by finding partial solutions with better conditional partial ratios. Since the feasible solution can be viewed as a composition of a sequence of partial solutions from the empty set, the approximation ratio is related to the conditional partial ratio of each partial solution.

In Theorem 1, the approximation ratio is upper-bounded by the maximum conditional partial ratio, while some building-block partial solutions may have lower conditional partial ratios but are not utilized. Moreover, in Theorem 1 we restrict SEIP to append partial solutions, while GSEIP can also remove partial solutions using bit-wise mutation. The approximation ratio can have a tighter bound if we consider these two issues. Applying the same principle as for Theorem 1, we present a theorem for GSEIP in particular that leads to a tighter approximation ratio.

Definition 6 *(Non-negligible path).* Given a minimization problem $(n, f, \mathcal{C})$ and an isolation function μ mapping to subsets of $[q]$, assume that every solution is encoded in an m-length binary vector. A set of solutions N is a *non-negligible path* with *ratios* $\{r_i\}_{i=0}^{q-1}$ and *gap* c if $\boldsymbol{x}^{\emptyset} \in N$ and, for every solution $\boldsymbol{x} \in N$, there exists a solution $\boldsymbol{x}' = (\boldsymbol{x} \cup \boldsymbol{y}^+ - \boldsymbol{y}^-) \in N$, where the pair of solutions $(\boldsymbol{y}^+, \boldsymbol{y}^-)$ satisfies

1. $1 \leqslant |\boldsymbol{y}^+| + |\boldsymbol{y}^-| \leqslant c$,
2. $f(\boldsymbol{y}^+ - \boldsymbol{y}^- \mid \boldsymbol{x}) \leqslant r_{|\mu(\boldsymbol{x})|} \cdot OPT$,
3. if $|\mu(\boldsymbol{x})| < q$, $|\mu(\boldsymbol{x} \cup \boldsymbol{y}^+ - \boldsymbol{y}^-)| > |\mu(\boldsymbol{x})|$.

Theorem 2. *Given a minimization problem* $(n, f, \mathcal{C})$ *and an isolation function* μ *mapping to subsets of* $[q]$, *assume that every solution is encoded in an m-length binary vector. If there exists a non-negligible path with ratios* $\{r_i\}_{i=0}^{q-1}$ *and gap c, then GSEIP finds a* $(\sum_{i=0}^{q-1} r_i)$-*approximate solution in expected time* $O(q^2 m^c)$.

Proof. We prove the theorem by tracking the process of GSEIP over the non-negligible path. We denote by $\boldsymbol{x}^{cur}$ the solution we want to operate on. Initially, $\boldsymbol{x}^{cur} = \boldsymbol{x}^{\emptyset}$, and thus $|\mu(\boldsymbol{x}^{cur})| = 0$.

GSEIP takes at most $O(q)$ expected steps to operate on $\boldsymbol{x}^{cur}$, since there are at most $q + 1$ solutions in the population and GSEIP selects one to operate on in each step.

According to the definition of the non-negligible path, there exists a pair of solutions $(\boldsymbol{y}^+, \boldsymbol{y}^-)$ with respect to $\boldsymbol{x}^{cur}$ such that $1 \leqslant |\boldsymbol{y}^+| + |\boldsymbol{y}^-| \leqslant c$. We denote $\boldsymbol{x}' = \boldsymbol{x}^{cur} \cup \boldsymbol{y}^+ - \boldsymbol{y}^-$. The probability that the mutation operator generates solution $\boldsymbol{x}'$ is at least $(\frac{1}{m})^c (\frac{m-1}{m})^{l-c}$, which implies $O(m^c)$ expected steps.

According to the definition of the non-negligible path, suppose that $|\mu(\boldsymbol{x}^{cur})| = i$; we also have $f(\boldsymbol{y}^+ - \boldsymbol{y}^- \mid \boldsymbol{x}^{cur}) \leqslant r_i \cdot OPT$. Note that $f(\boldsymbol{x}')$ can be decomposed recursively, and thus according to the theorem conditions, we have

$$f(\boldsymbol{x}') = f(\boldsymbol{y}^+ - \boldsymbol{y}^- \mid \boldsymbol{x}^{cur}) + f(\boldsymbol{x}^{cur})$$
$$= r_i \cdot OPT + f(\boldsymbol{x}^{cur}) = \cdots$$
$$= \sum_{j=0}^{i} r_j \cdot OPT + f(\boldsymbol{x}^{\emptyset}) = \sum_{j=0}^{i} r_j \cdot OPT.$$

Let $\mathcal{L}$ be a corresponding partial reference function. Thus, p-ratio$(\boldsymbol{x}') = \frac{\sum_{j=0}^{i} r_j \cdot OPT}{\mathcal{L}(\mu(\boldsymbol{x}'))}$.

Given $|\mu(\boldsymbol{x}^{cur})| = i$, again according to the definition of the non-negligible path, we have $|\mu(\boldsymbol{x}')| > |\mu(\boldsymbol{x})|$. Then we store solution $\boldsymbol{x}'$ in the population; otherwise there exists another solution $\boldsymbol{x}''$ with $|\mu(\boldsymbol{x}'')| = |\mu(\boldsymbol{x}')|$ and $\boldsymbol{x}''$ has a smaller partial ratio than $\boldsymbol{x}'$ by Lemma 1 when we substitute $\boldsymbol{x}''$ for $\boldsymbol{x}'$.

Now let $\boldsymbol{x}^{cur}$ be $\boldsymbol{x}'$. We have p-ratio$(\boldsymbol{x}^{cur}) \leqslant \frac{\sum_{j=0}^{|\mu(\boldsymbol{x}^{cur})|-1} r_j \cdot OPT}{\mathcal{L}(\mu(\boldsymbol{x}'))}$.

After at most q iterations of the above update of $\boldsymbol{x}^{cur}$, we have $|\mu(\boldsymbol{x}^{cur})| = q$, which means $\boldsymbol{x}^{cur}$ is feasible. Thus, the partial ratio of $\boldsymbol{x}^{cur}$, p-ratio$(\boldsymbol{x}^{cur}) = \frac{\sum_{j=0}^{q-1} r_j \cdot OPT}{\mathcal{L}(\mu(\boldsymbol{x}^{cur}))} = \frac{\sum_{j=0}^{q-1} r_j \cdot OPT}{OPT} = \sum_{j=0}^{q-1} r_j$, is its approximation ratio.

Thus, at most q jumps are needed to reach a feasible solution, each takes $O(m^c)$ expected steps for operation on a particular solution, and it takes $O(q)$ expected steps to choose the particular solution. Overall, it takes $O(q^2 m^c)$ expected steps to achieve the feasible solution. $\square$

Using Theorem 2 to prove the approximation ratio of GSEIP for a specific problem, we need to find a non-negligible path and then calculate the conditional evaluation function for each jump on the path. One way of finding a non-negligible path for a problem is to follow an existing algorithm for the problem. This will lead to a proof that the EA can achieve the same approximation ratio by simulating the existing algorithm. Similar ideas have been used to confirm that EAs can simulate dynamic programming [6]. In addition, note that the concrete form of the partial reference function is not required in the theorem.

5. SEIP for the MSCP

To apply SEIP to the MSCP, we use the fitness function

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x},$$

which has the objective of minimizing the total weight. For a solution $\boldsymbol{x}$, we denote $R(\boldsymbol{x}) = \bigcup_{S \in \boldsymbol{x}(C)} S$, that is, $R(\boldsymbol{x})$ is the set of elements covered by $\boldsymbol{x}$. We use the isolation function

$$\mu(\boldsymbol{x}) = R(\boldsymbol{x}),$$

which, owing to the effect of the isolation function, makes two solutions compete only when they cover the same number of elements. We could regard a partial reference function $\mathcal{L}$ of $\boldsymbol{x}$ to be the minimum price that optimal solutions pay for covering the same number of elements covered by $\boldsymbol{x}$, although it is not necessary to calculate the partial reference function.

Instead of directly assessing a minimum k-set cover problem $(n, \boldsymbol{w}, C, U, k)$, we analyze EAs for the *extended input* $(n, \boldsymbol{w}', C', U, k)$ [13]. The original problem is extended by first taking a closure of C under the subset operation, that is, $C' = C \cup \{2^S \mid \forall S \in C\}$, and the weight vector $\boldsymbol{w}'$ is extended accordingly by $w'(S) = \min\{w'(S_1), w'(S_2), \ldots, w'(S_j)\}$ if $S \subseteq S_1, S_2, \ldots, S_j$. Then if an optimal solution contains a set with less than k elements, we construct a new problem instance in which to U are added a minimum number of dummy elements such that all sets of the optimal solution are filled to be k-sets using dummy elements while keeping their weights unchanged. Therefore, the extended problem has an optimal solution containing k-sets. Analysis on the extended input leads to the same result as for the original problem, as shown in Lemma 3. The lemma is derived from Lemmas 2 and 3 of [13].
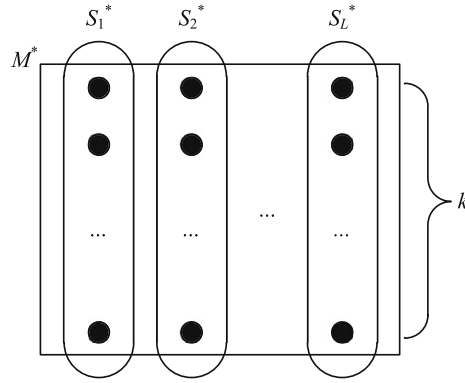
Fig. 1. The matrix M^* representation [13] of elements for the minimum k-set cover problem, containing exactly k rows.

Lemma 3. _The extended input does not affect the optimal cost or the optimal solutions. We can then assume without loss of generality that an optimal solution $\boldsymbol{x}^{OPT}$ consists of k-sets $\{S_1^*, S_2^*, \ldots, S_L^*\}$ and is disjoint, that is, $S_i^* \cap S_j^* = \emptyset$ for all i and j and $|S_i^*| = k$ for all i._

Thus, an optimal solution can be represented as a matrix M^* of elements, as plotted in Fig. 1, where column i corresponds to the elements in S_i^*. Note that there are exactly k rows in M^*, since each set in an optimal solution contains exactly k elements. For an element e, we denote $M^*(e)$ as the column to which e belongs, that is, the set in $\boldsymbol{x}^{OPT}$ that contains e. We also denote $w^*(e)$ as the cost of $M^*(e)$, and $N(e)$ as the number of uncovered elements in column $M^*(e)$ at the time at which e is covered.

5.1. SEIP ratio for the (unbounded) MSCP

In Theorem 3 we show that SEIP achieves an H_k-approximation ratio, where k is the cardinality of the largest set. It has been proved that SEMO [11] achieves an H_n-approximation ratio for the MSCP, which is known as the asymptotic lower bound for the problem. The theorem confirms that SEIP can simulate multi-objective EAs in terms of the solution retained, so that SEIP is able to achieve the approximation ratio obtained by multi-objective EAs.

Theorem 3. _Given an MSCP $(n, \boldsymbol{w}, C, U)$ where $|C| = m$, GSEIP finds an H_k-approximate solution in expected time $O(mn^2)$, where k is the size of the largest set in C._

The theorem is proved by simulating the greedy algorithm and using the property of the greedy algorithm as in Lemma 4, which is derived from [5].

Lemma 4. _Given an MSCP $(n, \boldsymbol{w}, C, U)$ and an arbitrary partial solution $\boldsymbol{x}$, let $\hat{S} = \arg\min_S r_S$ with respect to $\boldsymbol{x}$. For every element $e \in \hat{S}$, there exists a set $M^*(e)$ of an optimal solution that covers e, and it holds that,_

$$price(e) \leqslant \frac{w(M^*(e))}{|M^*(e) - R(\boldsymbol{x}^{cur})|}.$$

Proof of Theorem 3. We find a non-negligible path following the greedy rule; given the current partial solution, add the set $\hat{S}$ with minimum $r_{\hat{S}}$, as in Algorithm 1.

We denote $\boldsymbol{x}^{cur}$ as the current solution to be operated on. We find $\boldsymbol{y}^+ = \{\hat{S}\}$ where the set $\hat{S}$ minimizes $r_{\hat{S}}$ with respect to $\boldsymbol{x}^{cur}$. Let $\boldsymbol{y}^- = \emptyset$. Thus, we have $|\boldsymbol{y}^+| + |\boldsymbol{y}^-| = 1$ and $|\mu(\boldsymbol{x} \cup \boldsymbol{y}^+ - \boldsymbol{y}^-)| \geqslant |\mu(\boldsymbol{x})| + 1$ for partial solution $\boldsymbol{x}$.

By Lemma 4, for all $e \in \hat{S} - R(\boldsymbol{x}^{cur})$, there exists a set $M^*(e)$ of an optimal solution that covers e, and suppose

$$price(e) \leqslant \frac{w(M^*(e))}{|M^*(e) - R(\boldsymbol{x}^{cur})|}.$$

In the worst case, $|\hat{S} - R(\boldsymbol{x}^{cur})| = 1$, that is, the added set only covers one uncovered element e. In this case, according to the definition of $price(e)$, we have $w(\hat{S}) = price(e)$. We then have

$$\begin{aligned} f(\boldsymbol{y}^+ - \boldsymbol{y}^- \mid \boldsymbol{x}^{cur}) &= f(\boldsymbol{x}^{cur} \cup \boldsymbol{y}^+) - f(\boldsymbol{x}^{cur}) \\ &= w(\hat{S}) \leqslant \frac{1}{|M^*(e) - R(\boldsymbol{x}^{cur})|} \cdot w(M^*(e)). \end{aligned}$$

Thus, we find a non-negligible path with gap 1 and sum of ratios

$$
\begin{aligned}
\sum_e \frac{1}{|M^*(e) - R(\boldsymbol{x}^{cur}(e))|} \cdot w\big(M^*(e)\big) &= \sum_{j=1}^{|\boldsymbol{x}^{OPT}|} \sum_{e \in S_j^*} \frac{1}{|M^*(e) - R(\boldsymbol{x}^{cur}(e))|} \cdot w\big(M^*(e)\big) \\
&= \sum_{j=1}^{|\boldsymbol{x}^{OPT}|} \sum_{e \in S_j^*} \frac{1}{|M^*(e) - R(\boldsymbol{x}^{cur}(e))|} \cdot w\big(S_j^*\big) \\
&\leqslant \sum_{j=1}^{|\boldsymbol{x}^{OPT}|} w\big(S_j^*\big) \cdot \sum_{i=1}^{k} \frac{1}{|M^*(e) - R(\boldsymbol{x}^{cur}(e))|} \\
&= \sum_{j=1}^{|\boldsymbol{x}^{OPT}|} w\big(S_j^*\big) \cdot H_k \\
&= H_k \cdot OPT,
\end{aligned}
$$

where $\boldsymbol{x}^{cur}(e)$ denotes the partial solution that will cover e in its next step, and k is the size of the largest set in C.

By Theorem 2, GSEIP finds an H_k-approximate solution. Note that the isolation function maps to at most n isolations, the non-negligible path has a constant gap of 1, and the solution is encoded in an m-length binary vector; thus, GSEIP takes expected time $O(mn^2)$. □

Note that in the proof of Theorem 3, with respect to $\boldsymbol{x}^{cur}$, we find $|\boldsymbol{y}^-| = 0$ and $|\boldsymbol{y}^+| = 1$. Thus, the proof can be adapted to LSEIP directly, as in Theorem 4.

Theorem 4. *Given an MSCP $(n, \boldsymbol{w}, C, U)$ where $|C| = m$, LSEIP finds an H_k-approximate solution in expected $O(mn^2)$ time, where k is the size of the largest set in C.*

5.2. Ratio for GSEIP for the minimum k-set cover problem

In this section, we prove in Theorem 5 that GSEIP achieves the same $(H_k - \frac{k-1}{8k^9})$-approximation ratio as GAWW (Algorithm 2) [13], the current best algorithm for the minimum k-set cover problem. This result reveals that when the problem is bounded, which is very likely in real-world situations, GSEIP can yield a better approximation ratio than in the unbounded situation. Since the greedy algorithm cannot achieve an approximation ratio lower than H_n, the result also implies that GSEIP has essential non-greedy behavior for approximations.

Theorem 5. *Given a minimum k-set cover problem $(n, \boldsymbol{w}, C, U, k)$, where $|C| = m$, we denote $R(\boldsymbol{x}) = \bigcup_{S \in \boldsymbol{x}} S$. GSEIP using $\mu(\boldsymbol{x}) = R(\boldsymbol{x})$ finds an $(H_k - \frac{k-1}{8k^9})$-approximate solution in expected time $O(m^{k+1}n^2)$.*

When applying the GAWW rule to select sets, we use Lemmas 5 and 6. Owing to the assignments of $price(\cdot)$, the total price of elements covered by X equals the cost of X. We say a set S is *last-covered* if $\forall e \in S: N(e) = 0$.

Lemma 5 *(Lemma 4 of [13]). In each step of GAWW, its partial solution H is disjoint.*

Lemma 6 *(Lemma 5 of [13]). In each step of GAWW, we can assume without loss of generality that the set added to X (i.e., S^* or $\tilde{Q}$) has no more than one element in common with every set in $\boldsymbol{x}^{OPT}$.*

Lemmas 7 to 10 are derived from Lemmas 8, 9 and 10 in [13] and are required for calculating the weights of sets that GSEIP selects following the GAWW rule.

Lemma 7. *Given a minimum k-set cover problem $(n, \boldsymbol{w}, C, U, k)$ and an arbitrary partial solution $\boldsymbol{x}$, if the GAWW rule uses a withdrawal step to add sets $\tilde{Q} = \{S_1, S_2, \ldots, S_l\}$ to $\boldsymbol{x}$ and withdrawal $\tilde{S} \in \boldsymbol{x}$, denoting $R = \bigcup_{S \in \tilde{Q}} S$, then*

$$
\sum_{S \in \tilde{Q}} w(S) \leqslant w(\tilde{S}) + \sum_{e \in R - \tilde{S}} \frac{w^*(e)}{N(e) + 1} - \sum_{e \in R - \tilde{S}} \frac{w^*(e)}{k^4}.
$$

Lemma 8. *Given a minimum k-set cover problem* $(n, \mathbf{w}, C, U, k)$ *and an arbitrary partial solution* $\mathbf{x}$, *if the GAWW rule selects a set S that is not last-covered to add to* $\mathbf{x}$ *(greedy step), and there is an element* $e' \in S$ *such that*

$$price(e') < w^*(e')\left(\frac{1}{L_i + 1} - \frac{1}{4k^5}\right),$$

then

$$w(S) \leqslant \sum_{e \in S} \frac{w^*(e)}{N(e) + 1} - \sum_{e \in S} \frac{w^*(e)}{8k^8}.$$

Lemma 9. *Given a minimum k-set cover problem* $(n, \mathbf{w}, C, U, k)$ *and an arbitrary partial solution* $\mathbf{x}$, *if the GAWW rule selects a set S that is not last-covered to add to* $\mathbf{x}$ *(greedy step), followed by another set* S', *and for all elements* $e \in S$ *for which*

$$price(e) \geqslant w^*(e')\left(\frac{1}{h_i + 1} - \frac{1}{4k^5}\right),$$

then

$$w(S) + w(S') \leqslant \sum_{e \in S} \frac{w^*(e)}{N(e) + 1} - \sum_{e \in S} \frac{w^*(e)}{8k^8}.$$

Lemma 10. *Given a minimum k-set cover problem* $(n, \mathbf{w}, C, U, k)$ *and an arbitrary partial solution* $\mathbf{x}$, *if the GAWW rule selects a set S that is last-covered to add to* $\mathbf{x}$ *(greedy step), then*

$$w(S) \leqslant \sum_{e \in S} \frac{w^*(e)}{N(e) + 1}.$$

Proof of Theorem 5. We find a path of isolations following the GAWW rule. Note that there are at most $n + 1$ isolations.

For every $\mathbf{x}^{cur}$ belonging to an isolation $|\mu(\mathbf{x}^{cur})|$ on the path, if the GAWW rule selects sets satisfying Lemmas 7, 8, 9 and 10, we find $\mathbf{y}^+$ containing the sets added and $\mathbf{y}^-$ containing the set withdrawn. Thus, $|\mathbf{y}^+| + |\mathbf{y}^-| \leqslant k + 1$.

Since the GAWW rule covers at least one uncovered element, we have $|\mu(\mathbf{x}^{cur} \cup \mathbf{y}^+ - \mathbf{y}^-)| > |\mu(\mathbf{x}^{cur})|$.

As long as no last-covered set is included, by Lemmas 7, 8 and 9, the partial ratio of $f(\mathbf{y}^+ - \mathbf{y}^- \mid \mathbf{x}^{cur}) = w(\mathbf{y}^+) - w(\mathbf{y}^-)$ is upper-bounded as

$$w(\mathbf{y}^+) - w(\mathbf{y}^-) \leqslant \sum_{e \in (\bigcup_{S \in \mathbf{y}^+}) - (\bigcup_{S \in \mathbf{y}^-})} \left(\frac{w^*(e)}{N(e) + 1} - \frac{w^*(e)}{8k^8}\right);$$

otherwise, when the GAWW rule selects a last-covered set using a greedy step, by Lemma 10, noting $\mathbf{y}^- = \emptyset$,

$$w(\mathbf{y}^+) - w(\mathbf{y}^-) \leqslant \sum_{e \in (\bigcup_{S \in \mathbf{y}^+}) - (\bigcup_{S \in \mathbf{y}^-})} \left(\frac{w^*(e)}{N(e) + 1}\right).$$

In the worst case, there are $1/k$ among all the elements that are last-covered. Therefore, we find a non-negligible path.

By Theorem 2, GSEIP finds a solution with approximation ratio

$$\frac{1}{OPT}\left(\sum_{i=1}^{k} \frac{OPT}{i} - \frac{k-1}{k}\frac{OPT}{8k^8}\right) = H_k - \frac{k-1}{8k^9},$$

in expected time $O(m^{k+1}n^2)$. $\quad\square$

5.3. Comparison of the greedy algorithm, LSEIP and GSEIP

We assess the greedy algorithm, LSEIP and GSEIP for a subclass of the minimum k-set cover problem, denoted as problem I. We show in Propositions 1 to 3 that LSEIP and GSEIP can overcome the difficulty that limit the greedy algorithm, and thus yield better approximation ratios.

Problem I is a minimum k-set cover problem $(n, \mathbf{w}, C, U)$ constructed as follows. Note that $n = kL$ for some integer L. The optimal solution consists of L non-overlapping sets $\{S_i^*\}_{i=1}^{L}$, each of which contains k elements in U. Imagine that elements in each S_i^* are ordered and let S_{ij} contain only the jth element of set S_i^*. The collection of sets C consists of all sets of S_i^* and S_{ij}. Assign each S_i^* weight $1 + \epsilon$ for some $\epsilon > 0$, and assign each S_{ij} weight $1/j$. Problem I thus constructed is shown in Fig. 2.
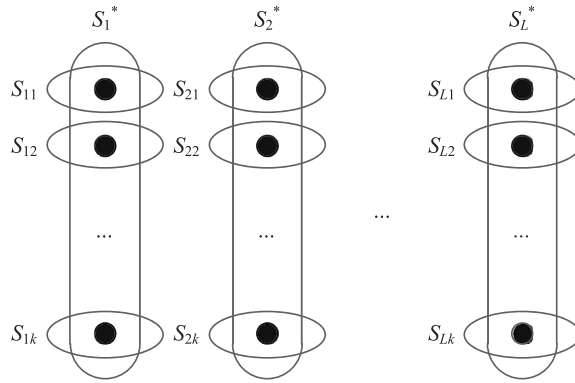
Fig. 2. Subclass I of the minimum k-set cover problem.

Proposition 1. *Given an arbitrary value $\xi > 0$, the approximation ratio of the greedy algorithm for problem I is lower-bounded by $H_k - \xi$.*

Proof. Note that the greedy algorithm adds the set with the minimum cost to the solution at each step. On initialization, the cost of S_i^* for all i is $\frac{1+\epsilon}{k}$, which is higher than the cost $\frac{1}{k}$ of S_{ik}, the smallest cost. Thus, S_{ik} for all i will be added to the solution. Then the cost of S_i^* for all i is $\frac{1+\epsilon}{k-1}$, which is higher than $\frac{1}{k-1}$ for $S_{i(k-1)}$. The greedy algorithm continues to choose a non-optimal set. Finally, all sets S_{ij} will be added to the solution. The approximation ratio is therefore $H_k - H_k \cdot \frac{\epsilon}{1+\epsilon}$. Let $\xi = H_k \cdot \frac{\epsilon}{1+\epsilon}$. ξ can be an arbitrarily small positive value as ϵ can be arbitrarily small. $\square$

Proposition 2. *With probability of at least $1 - \frac{1}{k+1}$, LSEIP finds an $(H_k - \frac{k}{n}(H_k - 1))$-approximate solution for problem I in $O((1 + \frac{1}{k})n^3)$ expected steps.*

Proof. Let LSEIP run until there are k elements uncovered by any solution identified. This takes $O(mn^2)$ expected steps by Theorem 4, which is $O((1 + \frac{1}{k})n^3)$ since $m = (1 + \frac{1}{k})n$. The uncovered k elements would be covered by sets in $\{S_i^*\}$. In the worst case, we assume the k elements are in one set $S_{\hat{i}}^*$ for $\hat{i}$.

Given a solution to be operated on, $S_{\hat{i}}^*$ is selected with probability $\frac{1}{m}$; set $S_{\hat{i}j}$ for $j \in \{1, 2, \ldots, k\}$ is selected with probability $\frac{k}{m}$; with the remaining probability, no more elements will be covered and we return to the solution. Therefore, to cover one of the k elements, $S_{\hat{i}}^*$ is selected with probability $\frac{1}{k+1}$; and set $S_{\hat{i}j}$ for $j \in \{1, 2, \ldots, k\}$ is selected with probability $\frac{k}{k+1}$.

Suppose one set $S_{\hat{i}\hat{j}}$ is selected. To cover one more uncovered element, $S_{\hat{i}}^*$ is selected with probability $\frac{1}{k}$; $S_{\hat{i}j}$ for $j \in \{1, 2, \ldots, \hat{j}, \ldots, k\}$ is selected with probability $\frac{k-1}{k}$. Therefore, the set $S_{\hat{i}}^*$ is selected with probability

$$\frac{1}{k+1} + \frac{1}{k}\left(1 - \frac{1}{k+1}\right) + \cdots + \frac{1}{2}\prod_{i=3}^{k+1}\left(1 - \frac{1}{i}\right)$$
$$= \frac{1}{k+1} \cdot \frac{k+1}{k+1} + \frac{1}{k} \cdot \frac{k}{k+1} + \cdots + \frac{1}{2} \cdot \frac{2}{k+1}$$
$$= \frac{k}{k+1} = 1 - \frac{1}{k+1},$$

and within $O((k+1)mn)$ steps, which is overwhelmed by $O(mn^2)$.

In the worst case, only one set S_i^* of an optimal solution is selected for the feasible solution. Thus, the approximation ratio is

$$\frac{(L-1)H_k + 1 + \epsilon}{L(1+\epsilon)} \leqslant \frac{L-1}{L}H_k + \frac{1}{L} = H_k - \frac{1}{L}(H_k - 1) = H_k - \frac{k}{n}(H_k - 1). \quad \square$$

Proposition 3. *GSEIP finds the optimal solution for problem I in $O(\frac{1}{k} \cdot (1 + \frac{1}{k})^{k+1} \cdot n^{k+3})$ expected steps.*

Proof. Let GSEIP run until a feasible solution is found, which takes $O(mn^2)$ expected steps according to Theorem 3. In the worst case, suppose the feasible solution found consists of all sets S_{ij}, so that the approximation ratio is $H_k - \xi$ for an arbitrary small value $\xi > 0$.

Keep GSEIP running until an optimal solution is found. GSEIP chooses to operate on the feasible solution with probability of at least $\frac{1}{n}$ as there are n isolations, which implies there are $O(n)$ steps.

Operating on the feasible solution, for some i, GSEIP uses its mutation operator to replace sets S_{ij} for all j and with S_i^* with probability $(\frac{m-1}{m})^{m-k-1}(\frac{1}{m})^{k+1}$, since $k+1$ bits in the solution are to be flipped, which implies there are $O(m^{k+1})$ steps. Once sets S_{ij} for all j are replaced with S_i^*, the partial ratio (approximation ratio) decreases and thus the mutated solution is retained.

Therefore, GSEIP takes $O(mn^2 + nm^{k+1}L)$ steps in all; note that $O(mn^2 + nm^{k+1}L) = O(mn^2 + n^2m^{k+1}/k) = O(\frac{1}{k} \cdot (1 + \frac{1}{k})^{k+1} \cdot n^{k+3})$, since $m = n(1 + \frac{1}{k})$. □

For GSEIP, we can also derive Proposition 4, proof of which is similar to that for Proposition 3.

Proposition 4. *GSEIP finds an $(H_k - c\frac{k}{n}(H_k - 1))$-approximate solution for problem I in $O(mn^2 + cn^2m^{k+1}/k)$ expected steps, for $c = 0, 1, \ldots, \frac{n}{k}$.*

This proposition illustrates an interesting property: compared with the greedy algorithm, whose performance cannot be improved given extra time, SEIP always seeks better approximate solutions. Users can allocate more running time in a trade-off for better solutions. However, the solution quality may not be improved by an EA over a long time for practical HP-hard problems. Thus, users should not assume any useful relationship between time allocated and the approximate ratio.

6. Conclusion

We studied the approximation performance of an EA framework. SEIP introduces an isolation function to manage competition among solutions, which can be configured as a single- or multi-objective EA. We analyzed the approximation performance of SEIP using the partial ratio and obtained a general characterization of the SEIP approximation behavior. Our analysis confirms that SEIP can achieve a guaranteed approximate solution: it tries to optimize the partial ratio of solutions in every isolation by finding good partial solutions, then these partial solutions form a feasible solution with a good approximation ratio.

We studied the performance of SEIP for the MSCP, which is NP-hard. Previous studies [11,28] showed that $(1 + 1)$-EA is not a good solver for MSCP, while a multi-objective EA achieves a similar approximation ratio to that of the greedy algorithm. Our analysis extends previous work to show that for the unbounded MSCP, SEIP achieves an H_k-approximation ratio (where H_k is the harmonic number of the cardinality of the largest set), the asymptotic lower bound. For the minimum k-set cover problem, it achieves an $(H_k - \frac{k-1}{8k^9})$-approximation ratio, the current best-achievable result [13], which is beyond the ability of the greedy algorithm. Moreover, for a subclass of the MSCP, we show how SEIP with either one-bit or bit-wise mutation can overcome the difficulty that limits the greedy algorithm.

We discussed some advantages and limitations of SEIP for approximations. To prove the SEIP approximation ratio for the MCSP problem, we used SEIP to simulate the greedy algorithm. Since the greedy algorithm is a general scheme for approximations and has been analyzed for many problems, SEIP analysis can be extended to cases for which the greedy algorithm has been applied, and therefore we can easily show that SEIP is a $\frac{1}{k}$-approximation algorithm for k-extensible systems [21], including b-matching, maximum profit scheduling and maximum asymmetric TSP problems. Moreover, to prove the approximation ratio of SEIP for the minimum k-set cover problem, we used SEIP to simulate the GAWW algorithm, which implies that SEIP also has extra behaviors that provide opportunities to exceed the greedy algorithm. However, limitations of SEIP are found from Theorem 2. There are some situations in which SEIP may fail. SEIP is required to flip a number of bits at a time, which depends on n, to achieve a good solution. In this situation, to flip the required number of bits, one-bit mutation is limited since it only flips one bit at a time, and bit-wise mutation is also limited as it requires exponential time. Further designs for elegant mutation operators may be possible. However, since the solution space has size 2^n, any mutation has to assign an exponentially small probability to some distant mutations. Once these distant mutations are required, exponential time is required. Another view of this limitation is that since the mutation operator only allows limited steps, SEIP may also over-optimize partial ratios, just like the greedy algorithm over-optimizes each step.

Our theoretical analysis suggests that EAs can achieve solutions with guaranteed performance. We believe that guaranteed and better approximation performance can be achieved by better EA design in the future.

Acknowledgements

References

[1] N. Alon, D. Moshkovitz, S. Safra, Algorithmic construction of sets for k-restrictions, ACM Transactions on Algorithms 2 (2) (2006) 153–177.
[2] A. Auger, B. Doerr, Theory of Randomized Search Heuristics – Foundations and Recent Developments, World Scientific, Singapore, 2011.
[3] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.
[4] E. Benkhelifa, G. Dragffy, A. Pipe, M. Nibouche, Design innovation for real world applications, using evolutionary algorithms, in: 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 2009, pp. 918–924.
[5] V. Chvátal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235.
[6] B. Doerr, A.V. Eremeev, C. Horoba, F. Neumann, M. Theile, Evolutionary algorithms and dynamic programming, in: Proceedings of the 11th ACM Conference on Genetic and Evolutionary Computation Conference (GECCO'09), Montreal, Canada, 2009, pp. 771–778.
[7] S. Droste, T. Jansen, I. Wegener, A rigorous complexity analysis of the $(1 + 1)$ evolutionary algorithm for linear functions with boolean inputs, Evolutionary Computation 6 (2) (1998) 185–196.
[8] R. Duh, M. Fürer, Approximation of k-set cover by semi local optimization, in: Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'97), El Paso, TX, 1997, pp. 256–264.
[9] U. Feige, A threshold of $\ln n$ for approximating set cover, Journal of the ACM 45 (4) (1998) 634–652.
[10] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, C. Witt, Analyses of simple hybrid algorithms for the vertex cover problem, Evolutionary Computation 17 (1) (2009) 3–19.
[11] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, C. Witt, Approximating covering problems by randomized search heuristics using multi-objective models, Evolutionary Computation 18 (4) (2010) 617–633.
[12] O. Giel, I. Wegener, Evolutionary algorithms and the maximum matching problem, in: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, 2003, pp. 415–426.
[13] R. Hassin, A. Levin, A better-than-greedy approximation algorithm for the minimum set cover problem, SIAM Journal on Computing 35 (1) (2005) 189–200.
[14] J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms, Artificial Intelligence 127 (1) (2001) 57–85.
[15] J. He, X. Yao, An analysis of evolutionary algorithms for finding approximation solutions to hard optimisation problems, in: Proceedings of 2003 IEEE Congress on Evolutionary Computation (CEC'03), Canberra, Australia, 2003, pp. 2004–2010.
[16] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara, N. Otsu, Real-world applications of analog and digital evolvable hardware, IEEE Transactions on Evolutionary Computation 3 (3) (1999) 220–235.
[17] G.S. Hornby, A. Globus, D.S. Linden, J.D. Lohn, Automated antenna design with evolutionary algorithms, in: Proceedings of 2006 American Institute of Aeronautics and Astronautics Conference on Space, San Jose, CA, 2006, pp. 19–21.
[18] J.R. Koza, M.A. Keane, M.J. Streeter, What's AI done for me lately? Genetic programming's human-competitive results, IEEE Intelligent Systems 18 (3) (2003) 25–31.
[19] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, K. Deb, Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem, in: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN'02), London, UK, 2002, pp. 44–53.
[20] A. Levin, Approximating the unweighted k-set cover problem: Greedy meets local search, SIAM Journal on Discrete Mathematics 23 (1) (2008) 251–264.
[21] J. Mestre, Greedy in approximation algorithms, in: Proceedings of the 14th Annual European Symposium on Algorithms (ESA'06), Zurich, Switzerland, 2006, pp. 528–539.
[22] F. Neumann, M. Laumanns, Speeding up approximation algorithms for NP-hard spanning forest problems by multi-objective optimization, in: Proceedings of the 7th Latin American Symposium on Theoretical Informatics, Valdivia, Chile, 2006, pp. 745–756.
[23] F. Neumann, J. Reichel, Approximating minimum multicuts by evolutionary multi-objective algorithms, in: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN'08), Dortmund, Germany, 2008, pp. 72–81.
[24] F. Neumann, I. Wegener, Minimum spanning trees made easier via multi-objective optimization, in: Proceedings of the 7th ACM Annual Conference on Genetic and Evolutionary Computation Conference (GECCO'05), Washington DC, 2005, pp. 763–769.
[25] F. Neumann, I. Wegener, Can single-objective optimization profit from multiobjective optimization? in: J. Knowles, D. Corne, K. Deb (Eds.), Multiobjective Problem Solving from Nature – From Concepts to Applications, Springer, Berlin, Germany, 2007, pp. 115–130.
[26] F. Neumann, C. Witt, Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity, Springer-Verlag, Berlin, Germany, 2010.
[27] F. Neumann, J. Reichel, M. Skutella, Computing minimum cuts by randomized search heuristics, in: Proceedings of the 10th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'08), Atlanta, GA, 2008, pp. 779–786.
[28] P. Oliveto, J. He, X. Yao, Analysis of the $(1 + 1)$-EA for finding approximate solutions to vertex cover problems, IEEE Transactions on Evolutionary Computation 13 (5) (2009) 1006–1029.
[29] R. Raz, S. Safra, A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, in: Proceedings of the ACM Symposium on Theory of Computing (STOC'97), El Paso, TX, 1997, pp. 475–484.
[30] J. Scharnow, K. Tinnefeld, I. Wegener, Fitness landscapes based on sorting and shortest paths problems, in: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN'02), Granada, Spain, 2002, pp. 54–63.
[31] P. Slavík, A tight analysis of the greedy algorithm for set cover, Journal of Algorithms 25 (2) (1997) 237–254.
[32] C. Witt, Worst-case and average-case approximations by simple randomized search heuristics, in: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05), Stuttgart, Germany, 2005, pp. 44–56.
[33] Y. Yu, Z.-H. Zhou, A new approach to estimating the expected first hitting time of evolutionary algorithms, Artificial Intelligence 172 (15) (2008) 1809–1832.