

# Guided Pareto Local Search and its Application to the 0/1 Multi-objective Knapsack Problems

Abdullah Alsheddy\*

Edward Tsang†

\*School of Computer Science and Electronic Engineering, University of Essex  
Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom  
aalshe@essex.ac.uk

†School of Computer Science and Electronic Engineering, University of Essex  
Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom  
edward@essex.ac.uk

## Abstract

Pareto Local Search (PLS) is a generalization of the local search algorithms to handle more than one objective. In this paper, two variants of PLS are examined on the multiobjective 0/1 knapsack problems, compared with three well-known multiobjective EA algorithms, namely SPEA, SPEA2 and NSGA2. Furthermore, A Guided Local Search (GLS) based multiobjective optimization algorithm is proposed, the Guided Pareto Local Search (GPLS). GPLS shows the ability of GLS to set on top of PLS not only to help PLS to escape Pareto local optimal set, but also to enhance its convergence toward and spread over the true Pareto front. Experimental results have shown that PLS can produce results with a very good quality, and proven the effectiveness of the GPLS.

## 1 Introduction

The multiobjective optimization problem (MOP) concerns the optimization of two or more objectives simultaneously. Instead of searching for a global optimum solution as in single-objective optimization problem, the search in MOPs targets a set of solutions representing the optimal set of trade-offs between the objectives. This set is known as the Pareto front (PF). All solutions in the PF are nondominated by any other solution in the same set (i.e. all solutions are Pareto equivalent to each other).

Over the past decade, a number of metaheuristics have been proposed for approximating the PF in a single run, many of which are multiobjective evolutionary algorithms (MOEAs) [13]. Some reviews of MOEAs algorithms are in [3] [2]. The popularity of EAs techniques is attributed to its population-based nature which enables the finding of multiple optima simultaneously. Algorithms based on single-point-based metaheuristics such as Tabu Search and Simulated Annealing have also been suggested to tackle MOP problems, however, at less extent [19] [1] [5] [11] [7]. Instead of maintaining a population, most of these algorithms incorporates the archive concept to store the nondominated solutions discovered during the local search processes.

Quiet recently, the Pareto Local Search (PLS) method [28, 32] has been proposed as a very simple algorithm for containing multiobjective optimization problems. Its simplicity stems from the fact that PLS is a straightforward extension to local search algorithms, no parameter settings

Hamburg, Germany, July 13–16, 2009

is required for PLS, and it does not involve any aggregation of objectives. Generally, PLS maintains a set of potentially efficient solutions, called archive, and iteratively improves this set by exhaustively exploring all of its neighbourhood. The acceptance criterion depends on the notion of Pareto optimality: a solution is accepted only if it is nondominated by all solutions in the archive. Furthermore, PLS has a natural stopping condition that is when the neighbourhoods of all solutions in the archive have been explored.

Several variants of PLS have been proposed [28, 32, 16]. Differences between such variants are in the way of defining its basic components [27]: (1) the selection scheme that chooses the current solutions from the archive to explore their neighbours, (2) neighbourhood exploration strategy, fully vs. partially, (3) the archiving, e.g. bounded vs. unbounded, and (4) the stopping criterion.

The applications of PLS have shown its ability of obtaining a good approximation of the efficient set of nondominated solutions [28, 32]. Its performance can be further improved when it is coupled with an efficient initial population generator as a two-phase algorithm [29], or with another evolutionary algorithm forming a hybrid algorithm [26, 30]. However, a major drawback of the PLS is the slow convergence that makes its computational time much higher than other state-of-the-art techniques [28, 29]. Besides, one can anticipate that PLS, as a local search method, can also be trapped in local optima points. This would become more serious when a simple neighbourhood function is defined, or partial neighbourhood visiting strategy is used.

Metaheuristics [31] includes many techniques that help the basic local search avoiding or escaping local optima. In this paper, we show that this can also be applied to the PLS methods. In order to enhance the convergence of the PLS and guide it to escape local optima, we propose the use of the Guided Local Search (GLS) [20] which is a general, yet very successful metaheuristic. GLS is a penalty-based metaheuristic algorithm that can be superimposed on other local search algorithms, with the aim of guiding it to escape local optima by dynamically augmenting the given objective function with penalties. When the local search settles in a local optima, GLS penalizes some selected features of the candidate solution.

Similarly, GLS can also be adapted to sit on top of PLS algorithms, and then we obtain a method that we call Guided Pareto Local Search (GPLS). In GPLS, the notion of local optimal is replaced by Pareto local optimal set. GPLS employs a new penalization strategy that incorporates knowledge obtained from the Pareto local optimal set.

There has been no attempt to implement a multiobjective GLS to tackle multiobjective optimization problems, up to our knowledge. This is normally attributed to its single-point-search nature. Therefore, GPLS can be considered as the first attempt to extend the GLS to contain more than one objective problems.

The purpose of this paper is twofold, first of which is to examine the performance of two PLS algorithms on a set of multiobjective 0/1 knapsack problem instances [24]. Second, examining the convergence performance of the GPLS by making a direct comparison with PLS algorithms, as well as other well-known EA approaches, namely SPEA2 [24] and NSGA2 [3].

The paper is organized as follows: section 2 and 3 describe GLS and PLS algorithms. GPLS is introduced in section 4. The application of PLS and GPLS to the multiobjective 0/1 knapsack problem and the experimental method are discussed in section 5. Experimental results are given in section 6. Finally, we conclude with a discussion and future work, section 7.

## 2 Guided Local Search

Guided Local Search (GLS) is a general metaheuristic algorithm for optimisation [22]. GLS sits on top of local search with the purpose of escaping from local minima by using a penalty-based approach. The basic idea is to augment the objective function with penalties, which directs the search away from local optimal. When the local search settles in a local optimal, GLS penalizes some selected features of the candidate solution.

To apply GLS, the solution features need to be defined in order to distinguish between solutions with different characteristics. Each feature is associated with a cost to help GLS

**Algorithm 1** Pseudo-code for steepest Pareto Local Search ( $PLS_s$ )

---

```

{ generate an initial solution, and add it to the archive}
 $s_0 \leftarrow InitialSolution()$ ;  $mark(s_0) \leftarrow false$ 
 $archive \leftarrow s_0$ 
while  $\exists s \in archive$  such that  $mark(s) = false$  do
  {fully explore the neighbourhood of  $s$ , and update the archive with nondominant neighbours.}
  for all  $s' \in Neighbourhood(s)$  do
    {evaluate  $s$  using a given set of objectives.}
     $Evaluate(s, [g_1, \dots, g_k])$ 
    if  $s'$  is nondominated by any solution in  $archive$  then
       $UpdateArchive(s')$ 
    end if
  end for
   $mark(s) \leftarrow true$ 
end while
return  $archive$ 

```

---

choose features that have more influence on the cost function in order to penalize them.

GLS uses an augmented function  $h(s)$  in the search process instead of the main objective function,  $g(s)$ :

$$h(s) = g(s) + \lambda \sum_{i \in F} p_i * I_i(s) \quad (1)$$

In this formula,  $s$  is a candidate solution and  $\lambda$  is a parameter of the GLS algorithm.  $F$  refers to the set of all features.  $p_i$  is the penalty of feature  $i$  (each  $p_i$  is initialized to 0), and  $I_i(s)$  is an indicator which is equal to 1 only if  $s$  exhibits the feature  $i$ ; 0 otherwise.

The basic GLS procedure can be described as follow: starting from an initial solution, a local search algorithm is applied until it reaches a local optimal. GLS augments the cost function by adding penalties to selected features. The novelty of GLS is mainly in the way that it selects which features to penalize. The intention is to penalize “unfavourable features” when a local search settles in a local optimal. The utility of penalizing feature  $i$ , ( $util_i$ ) under a local optimal  $s^*$ , is defined as follows:

$$util_i(s^*) = I_i(s^*) \times \frac{c_i}{1 + p_i} \quad (2)$$

where  $c_i$  is the cost and  $p_i$  is the current penalty value of feature  $i$ . At a local optimal, the feature with the greatest  $util$  value will be penalized. When a feature is penalized, its penalty value is always incremented by 1.

### 3 Pareto Local Search

Before describing the PLS, we have to introduce the notions of dominance and Pareto local optimal. In single objective optimization problems, different solutions are compared by performing the logic operators  $>$ ,  $<$  and  $==$  on the corresponding single objective function values. Solutions in a multiobjective optimization problem can be compared in the same way by using the concepts of *Pareto equivalence* (i.e.  $==$ ) and *Pareto dominance* ( $>$  and  $<$ ). Assuming a maximization problem, a solution  $x$  *dominates* a solution  $y$  (i.e.  $x \succ y$ ) if and only if  $x$  is no worse than  $y$  in all objectives and  $x$  is strictly better than  $y$  in at least one objective. The solution  $x$  is said to be *Pareto equivalent* to  $y$  (i.e.  $x == y$ ) only if neither  $x$  dominates  $y$  nor  $y$  is dominated by  $x$ . In multiobjective combinatorial optimization, usually there is not only a single Pareto optimal solution, but a set of Pareto optima, called *Pareto global optimal set*. Similarly, a solution  $s$

is a *Pareto local optimal* when it is nondominated by any other solution in its neighbourhood. Consequently a *Pareto local optimal set* is the set of Pareto local optimal solutions with respect to the defined neighbourhood.

Two PLS versions are considered here. The first PLS (called here  $PLS_s$ ) was proposed by Paquete et al [28]. As given in Algorithm 1,  $PLS_s$  starts with a randomly generated solution which is added to the archive (i.e. the *Pareto local optimal set*). Then, a candidate solution is randomly chosen from the archive, and its neighbourhood is fully explored (i.e. steepest visiting strategy). Every nondominant neighbour becomes a candidate to be added to the archive if it is nondominated by all solutions in the archive. After examining the neighbourhood of the current solution, it is marked as *visited*.  $PLS_s$  stops when all the solutions in the archive are visited.

The application of  $PLS_s$  to biobjective Travelling Salesman Problem (TSP) reveals that a major weakness of  $PLS_s$  is its excessive demand for computational time, particularly when the underlying neighbourhood operator is large such as the 3-opt for TSP. One reason for that is the neighbourhood visiting strategy applied by  $PLS_s$ . A full exploration of the neighbourhood of a solution is much more timely, compared to other visiting strategies. Thus, in cases where the time budget is limited, other greedy visiting strategies would be very competitive, particularly for many objective (i.e. more than two) optimization problems. For this reason, another variant of PLS is introduced here, called  $PLS_g$ , that applies a first improvement, partial visiting strategy.

The second PLS ( $PLS_g$ ) is depicted in Algorithm 2. It is similar to  $PLS_s$  with only two differences: (1)  $PLS_g$  applies a greedy visiting strategy where a PLS accepts the first better neighbour, (2) a solution  $s$  in  $PLS_g$  becomes a *Pareto local optimal* either when all its neighbour are considered and none of which dominates  $s$ , the maximum number of fails  $maxFails$  is exceeded or a maximum number of explored solutions in the archive  $maxRestarts$ . The number of fails (*fails*) is incremented every time a neighbour is dominated by the current solution, and it is reset to zero every time a move occurs. Another stopping condition is controlled by another parameter called  $maxRestarts$  which limits the number of solutions in the archive that to be considered and their neighbourhood to be explored.

Without using *one* or *both* of the two parameters, the  $PLS_g$  might be unable to reach the *Pareto local optimal set* while being given a limited amount of time, and maintaining a large or unbounded size of archive. Therefore,  $maxFails$  and  $maxRestarts$  parameters define earlier stopping criteria and, then, a virtual *Pareto local optimal set*. In this case, GPLS can be utilized even within a limited amount of computational time.

## 4 Guided Pareto Local Search (GPLS)

The proposed GPLS extends the single-objective GLS algorithm to guide the PLS to escape (real or virtual) *Pareto local optima set*. This requires some modifications to the single-objective GLS. As outlined in Algorithm 3 The GPLS modifies the GLS in the following areas:

### 4.1 Pareto Local Search

Instead of single-objective local search algorithms, a Pareto local search is applied to contain multiobjective optimization problems. Actually, any PLS, including the two versions described in section 3, that returns a *Pareto local optimal set* can be applied here. In this paper,  $PLS_g$  is used due to the imposed limitation in the computational time.

### 4.2 Features and their Cost

Features depend entirely on solution representation. In multiobjective optimization problems, one should take into consideration two different scenarios:

- All objectives share the same feature set. However, the cost of a feature varies according to a particular objective. In order to define the cost of a feature in this case, the influence of the feature on each objective function has to be considered and modelled into a single cost

**Algorithm 2** Pseudo-code for greedy Pareto Local Search ( $PLS_g$ )

---

```

 $s_0 \leftarrow InitialSolution()$ 
 $archive \leftarrow s_0$ 
 $fails \leftarrow 0$ 
while  $\exists s \in archive$  such that  $mark(s) = false$  AND  $restarts < maxRestarts$  do
  {apply first-improvement visiting strategy to the exploration of the neighbourhood of  $s$ , and update the
  archive with nondominant neighbours.}
   $restart \leftarrow restart + 1$ 
  for all  $s' \in Neighbourhood(s)$  do
    Evaluate( $s, [g_1, \dots, g_k]$ )
    if  $s'$  dominates  $s$  then
       $s \leftarrow s'$ 
       $fails \leftarrow 0$ 
    else if  $s$  does not dominate  $s'$  then
      if  $s'$  is nondominated by any solution in  $archive$  then
        UpdateArchive( $s'$ )
      end if
    else
       $fails \leftarrow fails + 1$ 
      if  $fails \geq maxFails$  then
        {exit the for loop}
        break
      end if
    end if
  end for
   $mark(s) \leftarrow true$ 
end while
return  $archive$ 

```

---

function. Such models include, for example, (weighted) aggregating approaches and other general functions e.g. min, max or mean. Modelling feature's cost is problem-dependant and, thus, it is hard to develop a general model. Problems in this category include the multiobjective 0/1 knapsack problem. Algorithm 3 considers this scenario.

- A distinct feature set needs to be defined for each objective, and a cost is associated with each feature to describe its influence on the corresponding objective. At the penalization phase, features in all feature sets should be considered to pick one or more features to be penalized. An example of problems in this category is the Travelling Salesman Problems With Profits[4].

### 4.3 Penalization Scheme

The guidance strategy that the single-objective GLS employs, relies on penalizing some features exhibited by the recent local optimal. As mentioned earlier, the novelty of GLS is mainly in the way that it selects features to penalize. The objective is to penalize "bad features" when the local search settles in a local optimal. Two factors affect the utility of a feature, namely its cost and the frequency of penalizing this feature.

GPLS deals with a *Pareto local optimal set* instead of a local optimal. A straightforward penalization strategy is to evaluate the utility of all features exhibited by *any* nondominant solution

**Algorithm 3** Guided Pareto Local Search

---

*GPLS* ( $[g_1, \dots, g_k], \lambda, [I_1, \dots, I_M], [c_1, \dots, c_M]$ )

 {where  $g_j$ : the objective function  $j$ ,  $\lambda$ : lambda parameter,  $I_i$ : indicator function of feature  $i$ ,  $c_i$ : cost of feature  $i$ ,  $M$ : number of features}
**BEGIN**

{set all penalties to 0}

**for all**  $i \in [1 \dots M]$  **do**
 $p_i \leftarrow 0$ ;
**end for**
 {define augmented objective functions:  $h_1, \dots, h_k$ }
**for all**  $j \in [1 \dots k]$  **do**
 $h_j \leftarrow g_j + \lambda * \sum_{i \in F} p_i * I_i$ ;
**end for****repeat**
 Apply *PLS* using  $[h_1, \dots, h_k]$  in solution evaluation: *Evaluate*( $s$ )

 { At *Pareto local optimal set*, compute the utility of features.  $\gamma_i$  represents how many solutions in *archive* exhibit feature  $i$  }
**for all**  $i \in [1 \dots M]$  **do**
 $util_i \leftarrow I_i(\text{archive}) * (c_i * (\gamma_i / |\text{archive}|)) / (1 + p_i)$ ;
**end for**

{ penalize features with maximum utility }

**for all**  $i$  such that  $util_i$  is maximum **do**
 $p_i \leftarrow p_i + 1$ ;

 {unmark any solution in *archive* exhibits  $i$ }
**for all**  $s \in \text{archive}$  such that  $I_i(s) = 1$  **do**
 $mark(s) \leftarrow \text{false}$ 
**end for****end for****until** *StoppingCriterion***return** *archive*;**END**

in the archive, and penalize features with maximum utility. However, this simple strategy does not incorporate any information from the archived solutions. Important information that can be extracted from the archive, is how many nondominant solutions exhibit a particular feature. This is another factor that can be incorporated in the utility function. Taking feature's occurrences into consideration in selecting the feature to penalize would help to prevent the search from directing all effort to any particular region of the pareto front, and, therefore, leads to a better spread over the true pareto front. It also enhances the chance of escaping Pareto local optima set by penalizing more solutions, with the hope that features with high cost that are exhibited by only one or few solutions, will be removed either by the PLS or by future penalization.

An alternative penalization scheme that incorporates besides feature's cost  $c_i$  and penalty  $p_i$ , the number of solutions in the *Pareto local optimal set* that exhibit this feature ( $\gamma_i$ ) is introduced. The utility function (Eq.2) is redefined as follow:

$$util_i(\text{archive}) = I_i(\text{archive}) \times \frac{c_i \times (\gamma_i / |\text{archive}|)}{1 + p_i} \quad (3)$$

where *archive* is a *Pareto local optimal set*,  $I_i(\text{archive})$  indicates whether "at least" one solution

in the archive exhibits feature  $i$ , and  $|archive|$  is its size of nondominant solutions. The feature with the greatest  $util$  value will be penalized. When a feature is penalized, its penalty value is always increased by 1.

This unique utility function redefines the term “bad feature”. If a feature is not exhibited in the *Pareto local optimal* (indicated by  $I_i$ ), then the utility of penalizing it is 0. The higher the cost of this feature (the greater  $c_i$ ) and the more nondominated solutions exhibit it (the greater  $\gamma_i$ ), the greater the utility of penalizing it. Besides, the more times that it has been penalized (the greater  $p_i$ ), the lower the utility of penalizing it again.

Having penalized a feature, all solutions in the *archive* that exhibit this feature need to be marked as not *visited*, so as to be considered by the PLS in the next iteration.

## 5 Multiobjective 0/1 Knapsack Problem

In order to test the GPLS algorithm, it is applied to the multiobjective 0/1 knapsack problems (MOKP). This problem was first formulated and solved by Zitzler & Thiele [25] using Strength Pareto Evolutionary Algorithm (SPEA). They formulated the problem using  $m$  knapsacks where the task is to maximize the profits simultaneously for all  $m$  knapsacks whilst satisfying weight constraints. Since then the problem has become a standard benchmark that has been solved by many other researchers (e.g. [3] [15] [23]). Later, Zitzler *et al* [24] introduced an improved version of SPEA, namely SPEA2. The performance of SPEA2 was tested on some sets of multiobjective 0/1 knapsack problem, and was compared to SPEA and NSGA2 [3] – a well-known multiobjective EA. They demonstrated the superiority of SPEA2 to its predecessor, and its competitive performance to NSGA2.

In this study we select as benchmarks, the data sets from the SPEA2, SPEA and NSGA2 runs. There are nine problems altogether with 250, 500, and 750 items, each of which with various numbers of objectives (2, 3 and 4 objectives). SPEA was applied to all instances, where SPEA2 and NSGA2 were only tested on the 750 items instances. At the time of writing, the problems together with the raw results obtained in Zitzler & Thiele [25] and [24] are available from an Internet web-site<sup>1</sup>.

### 5.1 Problem Formulation

A multiobjective variant of the 0/1 knapsack problem (MOKP) can be described by a set of  $n$  items and a set of  $m$  knapsacks. Given the capacity of knapsack  $j$  ( $\zeta_j$ ), the profit from including item  $i$  in knapsack  $j$  ( $\rho_{ij}$ ), and the weight of item  $i$  according to knapsack  $j$  ( $\omega_{ij}$ ), the task in MOKP is to find a vector  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , where  $x_i$  describes whether item  $i$  is put in all the knapsacks, that aims to:

$$\text{maximize } f_j(x) = \sum_{i=1, \dots, n} \rho_{ij} x_i, \quad \forall j = 1, \dots, m \quad (4)$$

$$\text{subject to } \sum_{i=1, \dots, n} \omega_{ij} x_i \leq \zeta_j, \quad \forall j = 1, \dots, m \quad (5)$$

### 5.2 The Implementation of GPLS for 0/1 MOKP

The only attempt to apply GLS to the knapsack problems, the authors aware of, is reported in [8], where the GLS has been applied to a single-objective variant of the multidimensional knapsack problem.

In order to apply the GPLS to the MOKP, the following components need to be defined.

<sup>1</sup><http://www.tik.ee.ethz.ch/sop/download/supplementary/testProblemSuite/>

### 5.2.1 Repair Method

MOKP involves constraints that are to be handled. This can be achieved by defining a heuristic for repairing infeasible solutions. Several repair approaches have been proposed for this purpose [25] [12]. In [25], a straightforward extension to a repair method applied to the single-objective approach was proposed. The repair algorithm removes items from the solution step by step until all capacity constraints are fulfilled. The order in which the items are deleted is determined by the maximum profit-to-weight ratio per item.

However, this heuristic favours items with a maximum ratio on a single objective, despite its impact on other objectives. An alternative, fairer heuristic that considers the profit-to-weight ratios with respect to all objectives is proposed here. Instead of the using the maximum profit-to-weight ratio per item, we use the summation of the profits divided by the summation of weights of each item over all knapsacks:

$$\frac{\sum_{j=1,\dots,m} \rho_{ij}}{\sum_{j=1,\dots,m} \omega_{ij}} \quad (6)$$

This repair method works very well with PLS, and yields a slightly better performance over the former one.

### 5.2.2 PLS

Three main components need to be defined in order to apply the PLS to 0/1 multiobjective knapsack problem, first of which is solution representation. We represent each solution in the search space by a vector  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ , where  $x_i$  describes whether item  $i$  is put in all the knapsacks. The second component is the neighbourhood function that maps a solution to a set of candidates. We use a simple neighbourhood function, by which a new neighbour is obtained by performing a single flip of any value in the current vector. Since flipping an item from 1 to 0 (i.e. the item is removed) will not yield an improved solution, it is replaced by a random non-included item (i.e. flipping from 0 to 1). The last component of LS is the initial solution which can be generated randomly or heuristically. In this study, we use a heuristic initialization by including all items ( $x_i = 1$ ), and applying the proposed repair method (Eq.6).

### 5.2.3 Features and their Costs

A possible feature to consider for this problem is whether an item  $i$  is included. The importance of putting item  $i$  in the knapsacks varies between knapsacks (i.e. objectives). In order to define a single value that determines the cost of a feature (i.e. choosing an item) on all objectives, simply the average of the profit-to-weight ratio of an item over all objectives is applied. Therefore, a “bad feature” can be defined as including an item with a overall *low* profit-to-weight ratio (i.e. high weight-to profit ratio), and, then, The cost of feature  $i$  is defined as follow:

$$c_i = \frac{\sum_{j=1,\dots,m} \omega_{ij} / \rho_{ij}}{m} \quad (7)$$

The lower the average of the profit-to-weight ratio for the  $m$  objectives of feature  $i$ , the higher the cost of this feature and, thus, the more chance this feature is to be penalized.

## 6 Experimental Results

As mentioned earlier, the obtained results of the SPEA, SPEA2 and NSGA2 algorithms on the 0/1 MOKP with 250, 500 and 750 items, and two, three and four objectives [24] are used in this study. Similar to [24], the running lengths allowed for those algorithms are expressed in terms of number of evaluations (*maxEval*), and the size of the *archive* is limited as given in Table 1. Accordingly, we have adjusted the stopping condition for both the PLS and GPLS to be equal to that of other algorithms. For PLS, the algorithm stops when the number of evaluations exceeds

Table 1: The stopping criteria expressed in terms of the maximum number of evaluations  $maxEval$ 

$n$	$m$	$maxEval$	$ archive $	$maxFails$	$\lambda$
250	2	75000	150	10	20
250	3	100000	200	15	20
250	4	125000	250	15	20
500	2	100000	200	15	30
500	3	125000	250	20	30
500	4	150000	300	20	30
750	2	125000	250	20	40
750	3	150000	300	25	40
750	4	175000	350	25	40

$maxEval$ . Initial experiments showed that PLS does not settle at real *Pareto local optimal set* within the allowed time. Therefore, in order to test GPLS, a virtual *Pareto local optimal set* is applied by using  $maxFails$  and/or  $maxRestarts$  parameters, as described in Algorithm 2. Only  $maxFails$  is used for biobjective instances, while both parameters are used in three and four objectives instances. Besides, a parameter of GPLS that needs to be tuned is  $\lambda$ . The setting of these parameters are given in Table 1. The  $maxRestarts$  is set to equal 20% of the archive size of each instance. Finally, following [24], each algorithm in this study has been independently run for 30 times for each test instance.

Due to the nature of MOPs, multiple performance indexes should be used for comparing the performances of different algorithms [2], [32]. In our experiments, the following performance index is used:

- **Set Coverage (C-metric):** Let  $X$  and  $Y$  be two approximations to the *Pareto global optimal set* (i.e. Pareto front,  $PF$ ) of a MOP,  $C(X, Y)$  is defined as the percentage of the solutions in  $Y$  that are dominated by at least one solution in  $X$ . When  $C(X, Y) = 1$ , all solutions in  $Y$  are dominated by some solutions in  $X$ . On the other hand,  $C(X, Y) = 0$  means that all solutions in  $Y$  are nondominated by any solution in  $X$ .  $C(X, Y)$  is not necessarily equal to  $1 - C(Y, X)$ .
- **Distance from Representatives in the PF (D-metric):** Let  $P^*$  be a set of uniformly distributed points along the Pareto front  $PF$ , or an upper approximation of the  $PF$ . Let  $A$  be an approximation to the  $PF$ , the average distance from  $P^*$  to  $A$  is defined as follow:

$$D(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \quad (8)$$

where  $d(v, A)$  is the minimum Euclidean distance between  $v$  and the points in  $A$ . The lower the value of  $D(A, P^*)$ , the closer  $A$  to the  $PF$  will be. In this study, the upper approximation to each 0/1 knapsack test instance that was proposed in [ ] is used here as  $|P^*|$ .

Table 2 presents the means of the C-metric values of the final approximations obtained by both variants of PLS ( $PLS_s$  and  $PLS_g$ ) and  $GPLS$ , compared to SPEA on instances of size 250 and 500. The means of the C-metric for the PLS variants and  $GPLS$ , compared to SPEA2 and NSGA2 on instances of size 750 are given in Table 3. Table 4 gives the means and standard deviations of the values of D-metric for the  $PLS_s$ ,  $PLS_g$  and  $GPLS$ . Fig.2 shows the evolution of the D-metric value with the number of evaluations in each algorithm for the 750 test instances. From these tables and figures, the following remarks can be made:

- Both variants of Pareto Local Search are superior to the other MOEA algorithms in terms of the C-metric. They were capable to produce an approximation to the true Pareto front

Table 2: Average set coverage between the proposed algorithms (PLS variants and GPLS) and SPEA

$m$	$n$	$C(pls_s, spea)$	$C(spea, pls_s)$	$C(pls_g, spea)$	$C(spea, pls_g)$	$C(GPLS, spea)$	$C(spea, GPLS)$
250	2	0.83	0.0	0.83	0.0	<b>0.94</b>	0.0
250	3	0.83	0.0	0.84	0.0	<b>0.95</b>	0.0
250	4	<b>0.98</b>	0.01	0.92	0.0	0.69	0.0
500	2	<b>1.0</b>	0.0	0.99	0.0	<b>1.0</b>	0.0
500	3	0.89	0.0	0.89	0.0	<b>0.99</b>	0.0
500	4	<b>0.1</b>	0.0	0.89	0.0	<b>1.0</b>	0.0

Table 3: Average set coverage between the proposed algorithms (PLS variants and *GPLS*) and SPEA2 and NSGA2

$m$	$n$	$C(pls_s, spea2)$	$C(spea2, pls_s)$	$C(pls_g, spea2)$	$C(spea2, pls_g)$	$C(GPLS, spea2)$	$C(spea2, GPLS)$
750	2	0.49	0.0	0.13	0.0	<b>0.86</b>	0.0
750	3	0.75	0.0	0.75	0.0	<b>0.94</b>	0.0
750	4	0.87	0.0	0.86	0.0	<b>0.97</b>	0.0
$m$	$n$	$C(pls_s, nsga2)$	$C(nsga2, pls_s)$	$C(pls_g, nsga2)$	$C(nsga2, pls_g)$	$C(GPLS, nsga2)$	$C(nsga2, GPLS)$
750	2	0.69	0.0	0.67	0.0	<b>0.96</b>	0.0
750	3	0.73	0.0	0.74	0.0	<b>0.92</b>	0.0
750	4	0.83	0.0	0.82	0.0	<b>0.94</b>	0.0

with a very good quality. Table 2 and Table 3 show that the final set of solutions produced by  $PLS_s$  or  $PLS_g$  is better in all instances than those of other MOEA algorithms in terms of C-metric. Actually, none of the MOEA algorithms were able to generate a solution that dominate any solution in obtained by PLS variants.

- Within a limited amount of time, using the concept of *virtual Pareto local optimal set* and then applying the *GPLS* significantly enhances the performance of PLS in terms of solutions quality and diversity. This revealed in Table 2, 3 and 4, the first two tables show that the final set of solutions produced by  $PLS_s$  or  $PLS_g$  is improved by using the penalization approach in GPLS in all instances, except 250-4, in terms of C-metric. *GPLS* is also capable of improving the D-metric values significantly as given in Table 4. The

Table 4: Means (Standard deviations) of the D-metric values of  $PLS_s$ ,  $PLS_g$  and *GPLS*

$m$	$n$	$pls_s$	$pls_g$	<i>GPLS</i>
250	2	239.9(70.74)	247.6(70.9)	<b>177.77</b> (22.91)
250	3	771.51(30.56)	777.19(32.73)	<b>522.08</b> (23.88)
250	4	892.75(21.66)	916.17(19.48)	<b>755.43</b> (16.47)
500	2	512.79(41.57)	526.05(51.8)	<b>407.5</b> (51.84)
500	3	1949(44.3)	1959(39.1)	<b>1640.3</b> (53.1)
500	4	2400.5(30.6)	2432(38.45)	<b>2096</b> (43.61)
750	2	1285.3(52.3)	1298.2(56.5)	<b>845.3</b> (97.4)
750	3	3011.3(34)	3004(32.9)	<b>2611.9</b> (54.3)
750	4	4131.6(32.6)	4181.6(29.2)	<b>3695.5</b> (49.3)

difference between the approximations obtained by PLS variants, GPLS and an MOEA algorithm on instances 2502, 5002 and 7502 can be visually detected from Fig.1.

- Importantly, Fig.2 clearly indicates that for all number of objectives, *GPLS* needs fewer number of evaluations than PLS variants for minimizing the D-metric value, which suggests that *GPLS* is more efficient and effective than the PLS.

Overall, these remarks suggest that PLS is a very good technique even within a time limit constraint and many objectives. It is better than well known MOEA algorithms such as SPEA, SPEA2 and NSGA2 in the 0/1 knapsack problems. We can also claim that *GPLS*, in overall, is better than PLS in terms of convergence speed and solution quality, in this set of test instances.

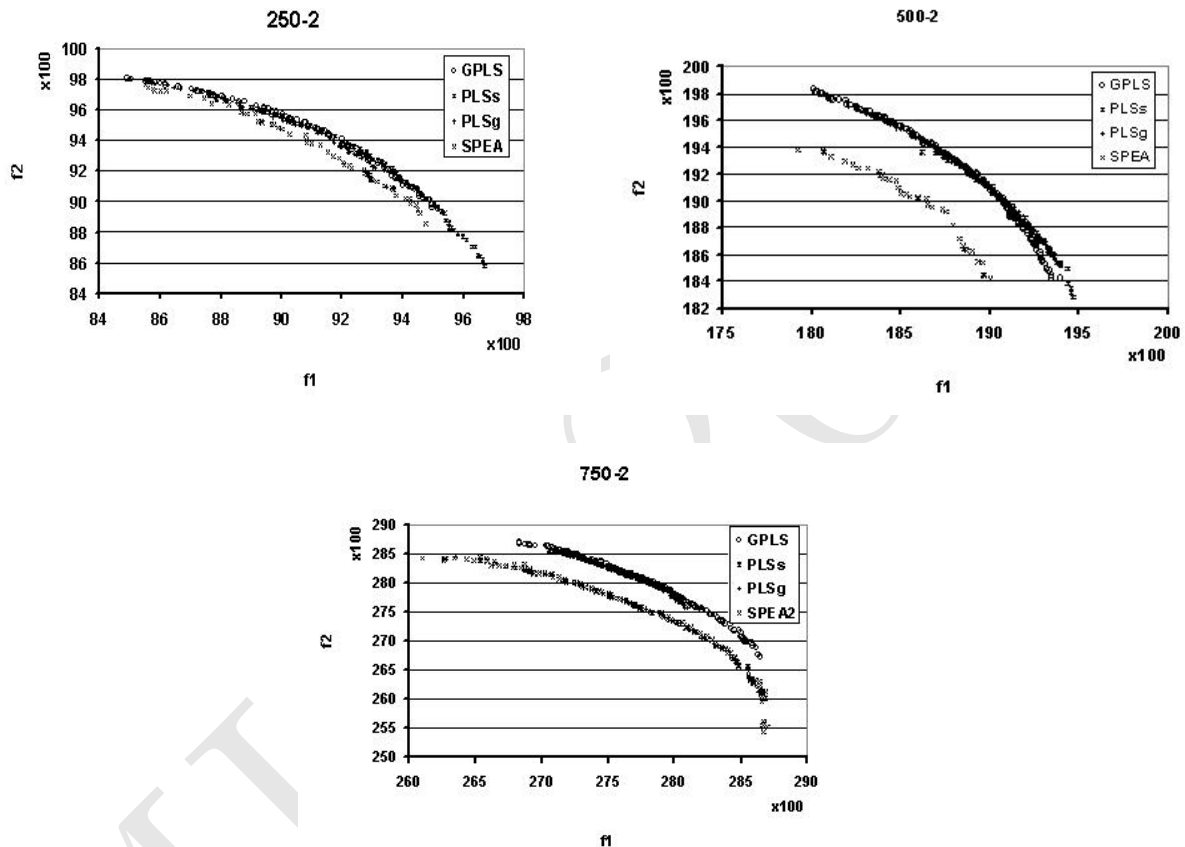


Figure 1: Plots of a nondominated solutions set of PLS variants, GPLS and an MOEA algorithm for all the 2-objective MOKP test instances.

## 7 Discussion and Conclusions

A multiobjective algorithm based on GLS (*GPLS*) for solving multiobjective optimization problems has been proposed. *GPLS* confirms the ability of GLS to be superimposed on Pareto local search algorithms in order to guide it to escape Pareto local optimal sets. Besides, *GPLS* can also help the PLS to improve its convergence and solution quality when the time is limited. *GPLS* slightly modifies the penalization scheme of the single-objective GLS, by incorporating a third factor (together with features' cost and penalty) in evaluating the utility of penalizing a

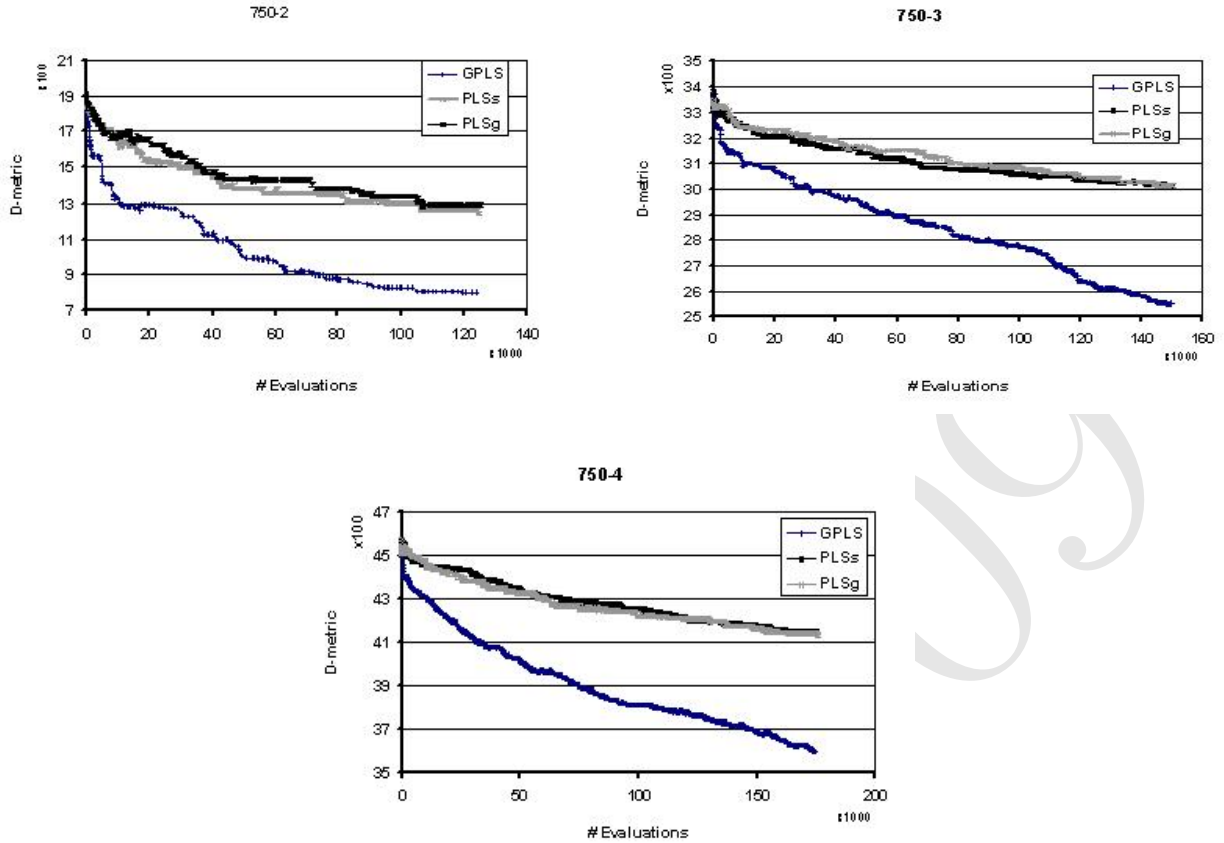


Figure 2: An example of the the evolution of the D-metric for PLS variants and GPLS

particular feature. This factor employs knowledge about the features of solutions in the *Pareto local optimal set*. The more nondominated solutions in the obtained *Pareto local optimal set* exhibit a particular feature, the greater the utility of penalizing it.

Given a limited amount of time, applying GPLS requires an early stopping condition of the underlying PLS, defining a virtual *Pareto local optimal set*. GPLS together with two variants of PLS have been applied to the multiobjective 0/1 knapsack problem. The results clearly show the effectiveness of PLS algorithms compared to three well known MOEAs. Moreover, the results confirm that GPLS enhances the performance of PLS in terms of convergence and solution quality.

Although GPLS has been applied on top of a variant of PLS, its generality is clearly described. Any PLS that returns a real or virtual *Pareto local optimal set*, the proposed GPLS can guide it in a straightforward manner.

According to recent studies [9] [10], Pareto ranking-based MOEAs such as NSGA2 do not work well on many objective optimization problems. This is an interesting and appealing feature of PLS and GPLS, as variants of pareto ranking-based methods, to obtain very good results even in many objective optimization problems. However, PLS and GPLS still outperformed by other state-of-art algorithms in the 0/1 MOKP such as MOEA/D [23] which is an aggregation based MOEA. The high convergence property of GPLS encourages for investigating designing a framework that embeds the GPLS into it, e.g. a parallel GPLS, to enhance the solution diversity in the objective space. This describes one further research that we are studying currently.

## References

- [1] S. Bandyopadhyay, S. Saha, U. Maulik and K. Deb. A simulated annealing based multi-objective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3):269-283, 2008.
- [2] C.A.C. Coello. Evolutionary multiobjective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28-36, 2006.
- [3] K. Deb, A. Pratab, S. Agrawal and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGAI. *IEEE Transactions on Evolutionary Computation*, 6(2):182-197, 2002.
- [4] D. Feillet, P. Dejax and M. Gendreau. Traveling Salesman Problems With Profits: An Overview. *Transportation Science*, 39:188-205, 2001.
- [5] X. Gandibleux and A. Freville. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objective case. *J. Heuristics*, 6(3):361-383, 2000.
- [6] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, New York, 1989.
- [7] M. P. Hansen. Tabu Search in Multiobjective Optimisation: MOTS. In T. Stewart and R. van den Honert, editors, *Proceedings of 13th International Conference on Multiple Criteria Decision Making*, Springer-Verlag, Berlin, 1996.
- [8] M. Hifi, M. Michrafy, and A. Sbihi. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. *Journal of the Operational Research Society*, 55:1323-1332, 2004.
- [9] E. J. Hughes. Evolutionary many-objective optimization: Many once or one many?. in *Proceedings of 2005 Congress on Evolutionary Computation, Edinburgh, Scotland, UK*, 2005.
- [10] H. Ishibuchi, T. Doi, and Y. Nojima. Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. In *Parallel Problem Solving from Nature IX (PPSN-IX)*, 4193:493-502, 2006.
- [11] D. Jaeggi, C. Asselin-Miller, G. Parks, T. Kipouros, T. Bell and J. Clarkson. Multi-objective Parallel Tabu Search. In *Parallel Problem Solving from Nature - PPSN VIII*, 732-741, Springer, Berlin, 2004.
- [12] A. Jaszkievicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402-412, 2002.
- [13] D. Jones, S. Mirrazavi, M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137:1-9, 2002.
- [14] P. Kilby, P. Prosser, and P. Shaw. Guided local search for the vehicle routing problem with time windows. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, 473-486, 1999.
- [15] J. D. Knowles and D. W. Corne. M-PAES: A memetic algorithm for multiobjective optimization. *Proceedings 2000 Congress on Evolutionary Computation Piscataway, NJ: IEEE Press*, 1:325-332, July 2000.
- [16] J. D. Knowles and D. W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8,2:149-172, 2000.
- [17] J. D. Schaffer. Multiple-objective optimization with vector evaluated genetic algorithms. In J. Grefenstette, editors, *Genetic Algorithms and Their Applications: Proceedings of the Third International Conference on Genetic Algorithms*, pages 93-100. Lawrence Erlbaum, Hillsdale, 1985.

- [18] E.P.K. Tsang and C. Voudouris. Fast local search and guided local search and their application to British Telecom's workforce scheduling problem. *Operations Research Letters*, 20(3):119–127, 1997.
- [19] E. L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *J. Multi-Criteria Decision Anal.*, 8(4):221–236, 1999.
- [20] C. Voudouris and E.P.K. Tsang. Guided Local Search for Combinatorial Optimisation Problems. PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, 1997.
- [21] C. Voudouris and E.P.K. Tsang. Guided Local Search and its application to the Travelling Salesman Problem. *European Journal of Operational Research*, 113(2):469–499, 1999.
- [22] C. Voudouris and E.P.K. Tsang. Guided local search. In: F. Glover, editors, *Handbook of meta-heuristics*, 185–218, Kluwer, 2003.
- [23] Q. Zhang, H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.
- [24] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, May 2001.
- [25] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999.
- [26] R. Kumar and P. K. Singh Pareto Evolutionary Algorithm Hybridized with Local Search for Biobjective TSP. *Hybrid Evolutionary Algorithms*, 361–398, 2007.
- [27] A. Liefoghe, S. Mesmoudi, J. Humeau, L. Jourdan, E. Talbi A Study on Dominance-Based Local Search Approaches for Multiobjective Combinatorial Optimization. *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, 120–124, 2009.
- [28] L. Paquete, M. Chiarandini and T. Stutzle Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study. *LECTURE NOTES IN ECONOMICS AND MATHEMATICAL SYSTEMS*, 535:177–200, 2004.
- [29] T. Lust and J Teghem Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 2009.
- [30] A. Jaszkiwicz and P. Zielniewicz Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research*, 193:3:885–890, March 2009.
- [31] F. Glover and G. Kochenberger Handbook of Metaheuristics (International Series in Operations Research & Management Science). *Springer* 2003.
- [32] E. Angel, E. Bampis and L. Gourves. A Dynasearch Neighborhood for the Bicriteria Traveling Salesman Problem. *LECTURE NOTES IN ECONOMICS AND MATHEMATICAL SYSTEMS*, 535:153–176, 2004.