

An Empirical Study of Tabu Search for the MOKP

Vincent Barichard and Jin-Kao Hao

LERIA - Faculty of Sciences - University of Angers

2 Boulevard Lavoisier

49045 Angers Cedex 01 FRANCE

email: {Vincent.Barichard, Jin-Kao.Hao}@info.univ-angers.fr

Abstract : Three Tabu Search algorithms are presented for the 0-1 multi-dimensional multi-objective knapsack problem. Experiments are carried out to study the role of different diversification techniques for TS and to compare TS with well-known evolutionary algorithms.

Keywords : Multi-objective optimization, Knapsack, Tabu Search, Diversification, Heuristics

1 Introduction

Given a set of items (or objects), each one being associated with a vector of profits and weights, the 0-1 multi-dimensional multi-objective knapsack problem (MOKP) consists in selecting a subset of items in order to maximize a multi-objective function while satisfying a set of knapsack constraints. More formally, the MOKP can be stated as follows:

$$\text{MOKP01} \left\{ \begin{array}{ll} \max & z^j(x) = \sum_{i=1}^n c_i^j x_i \quad j = 1, \dots, o \\ \text{s.t.} & \sum_{i=1}^n w_i^l x_i \leq b_l \quad l = 1, \dots, m \\ & x_i \in \{0, 1\} \quad i = 1, \dots, n \end{array} \right.$$

where n is the number of items, x_i is a decision variable o the number of objectives, z^j is the j^{th} component of the multi-objective function z , and m the number of knapsack constraints of the problem. MOKP is one of the most studied problems in the area of multi-objective combinatorial optimization.

Just like the NP-hard 0-1 single multi-dimensional knapsack problem MKP, the MOKP may be used to formulate many practical problems such as capital budgeting and resource allocation. For resolution purpose, several heuristic algorithms have been developed. We may mention neighborhood algorithms such as simulated annealing [14, 16] and Tabu Search (TS) [5], population based evolutionary algorithms such as “vector evaluated genetic algorithm” (VEGA) [12], “non-dominated sorting genetic algorithm” (NSGA) [15], “strength Pareto evolutionary algorithm” (SPEA) [18] and “memetic Pareto archived evolution strategy algorithm”

(*M-PAES*) [2]. Very recently, hybrid algorithms combining genetic search and local search as “multiple objective genetic local search” (*MOGLS*) [9] were also proposed.

Previous studies tend to show some advantages of population based evolutionary algorithms (EA) over neighborhood algorithms in the context of multi-objective combinatorial optimization. Indeed, given that solving a multi-objective problem requires finding a *set* of tradeoff solutions (the Pareto or near Pareto front), the concept of population used by EAs is naturally quite appealing. In such a situation, it is clear that diversification plays an important role for an efficiency search.

In this paper, we show that neighborhood algorithms without using a population is also able to effectively find trade-off solutions. We demonstrate experimentally that the key to achieve such an effectiveness is to incorporate a carefully guided diversification technique. To illustrate the point, we present three Tabu Search algorithms for the MOKP. The first algorithm (called *TS*) is a basic TS algorithm where diversification is ensured only by the tabu list. The second algorithm (*TS+RW*) integrates a random-walk mechanism to diversify the search. The last algorithm (*TS+HW*) uses an original measure to monitor the diversity evolution of some previously visited solutions. According to the measured degree of diversity, the algorithm decides whether it continues its search in the current direction (intensification) or goes to another search area (diversification).

To assess the effectiveness of these diversification techniques, we carry out experiments using some well-known MOKP benchmark instances. Also we compare our best TS algorithm with two state of the art hybrid evolutionary algorithms, we show that even if the TS algorithm operates with a single solution during its search, it may compete favorably in some cases with these population based evolutionary algorithms.

The paper is organized as follows: in the next section, we introduce the general principles of our TS algorithms. Experimental study as well as comparisons of *TS*, *TS+RW* and *TS+HW* are the subjects of the section 3. Then we compare the best Tabu algorithm with two state-of-the-art algorithms. Last section gives some conclusions and perspectives.

2 Tabu algorithms

Brief review of Tabu Search

Tabu Search is an advanced neighborhood search method with a set of critical and complementary components [6]. For a given instance of an optimization problem characterized by a search space S and a cost function f , a function $N : S \rightarrow 2^S$ is first introduced to determine a neighborhood. A typical TS algorithm begins then with an initial configuration x in S and then proceeds iteratively to visit a series of locally best configurations following the neighborhood function. At each iteration, a *best* neighbor $x' \in N(x)$ is sought to replace the current configuration even if x' does not improve the current configuration in terms of the cost function. To avoid the problem of possible cycling and to allow the search to go beyond local optima, TS introduces the notion of *Tabu List*, one of the most important components of the method.

A tabu list is a special short term memory that maintains a selective history H , composed of previously encountered solutions or more generally pertinent attributes of such solutions. A simple TS strategy based on this short term memory H consists in preventing solutions of H from being reconsidered for next k iterations (k , called tabu tenure, is problem dependent).

When attributes of solutions instead of solutions are recorded in tabu list, some non-visited, yet interesting solutions may be prevented from being considered. *Aspiration criteria* may be used to overcome this problem. One widely used aspiration criterion consists of removing a tabu classification from a move when the move leads to a solution better than the best obtained so far.

An efficient TS algorithm must establish a good compromise between exploration and exploitation during its search. This is ensured by mechanisms called “diversification and intensification”. We show below diversification is particularly important for multi-objective optimization.

The common features of our Tabu algorithms

Now, we introduce the common features used by the three Tabu algorithms.

Search space. A configuration is any binary vector $x = (x_1, x_2, \dots, x_n)$ verifying all the knapsack constraints, *i.e.* $\forall l \in \{1, \dots, m\}, \sum_{i=1}^n w_i^l x_i \leq b_l$. The search space S is then composed of all such vectors, *i.e.* $S = \{x \in \{0, 1\}^n \mid \sum_{i=1}^n w_i^l x_i \leq b_l, l = 1, \dots, m\}$.

Neighborhood. The neighborhood N of our problem is defined as follows. Given $x, x' \in S$, x and x' are neighboring ($x' \in N(x)$) if they differ by the value of a single variable. More formally, $N(x) = \{x' \in S \mid \text{hammingdistance}(x, x') = 1\}$. Therefore, one can get a neighboring configuration x' by adding or dropping one item ($x_i : 0 \rightarrow 1$ or $1 \rightarrow 0$) from x such that x' remains feasible. The move from x to x' is then characterized by an integer i , the index of the flipped variable x_i . This index is called the attribute of the move.

Evaluation of the neighborhood. The evaluation of the neighborhood is based on the so-called weighted Tchebycheff approach [10, 4, 3]¹. Formally, given $x = (x_1, x_2, \dots, x_n) \in S$ and $x' = (x'_1, x'_2, \dots, x'_n) \in N(x)$, the evaluation of x' is defined by the following function:

$$\text{eval}(x, x') = \min_j \lambda_j (z^j(x') - z^j(x))$$

where

- $z^j(x') = \sum_{i=1}^n c_i^j \times x'_i$, the value of the j^{th} objective of x' ;
- λ_j , a component of the vector $\lambda \in [0, 1]^o$, where $(\sum_{j=1}^o \lambda_j) = 1$. The vector λ corresponds to a direction in the objectives space, and allows a diversification of the search.

Choice of the best neighbor. Let x the current configuration of the algorithm, the neighboring configuration m to replace x is determined thanks to the following equation: $\text{eval}(x, m) = \max_{x' \in N(x)} \text{eval}(x, x')$. If several configurations verify the equation, the algorithm will randomly choose one of them.

Management of the Tabu list. Each time a move is carried out to go from x to x' , the index of the flipped variable is recorded in the tabu list. Therefore, the reverse move is forbidden for next k iterations of the algorithm. To implement the tabu list, we use an n array T . Each time an index is added to the tabu list, the corresponding element of T is set to the current iteration number plus tabu tenure k . At any moment, it is easy to verify if a given move is tabu or not by simply comparing the current iteration number with that recorded in the tabu array T .

The Aspiration criterion. The above tabu mechanism is sufficient to prevent the algorithm from being trapped in short-term cycling. At the same time, such a mechanism may forbid configurations that are not yet visited. To overcome this problem, a standard and simple aspiration criterion is introduced. A tabu move is chosen if the move leads to a configuration whose evaluation is better than that of the best configuration found so far by the algorithm.

Characteristics of the three Tabu algorithms

Now we may introduce our three Tabu algorithms: the basic tabu algorithm (*TS*), the random walk tabu algorithm (*TS+RW*) and the tabu algorithm based on hamming distance (*TS+HW*).

The basic algorithm *TS*. The basic tabu algorithm *TS* results directly from the outline presented in section 2 (see algorithm 1). The diversification is ensured only by the tabu list.

The random walk tabu algorithm *TS+RW*. The principle of random walk (RW) is well-known in the literature and

¹Two other popular approaches are based on aggregation [1] and ranking [7].

In: a feasible configuration x_{init} , the number of iterations L
Out: a new feasible configuration x

```

 $x \leftarrow x_{init}$ 
for  $i = 0$  to  $L$  do
  choose the best authorized move  $m$ 
  update the tabu list with  $m$ 
  perform the chosen move  $m$  in  $x$ 
  update the set of non-dominate configurations with  $x$ 
end for

```

Algorithm 1: The basic Tabu search algorithm TS .

used in some famous algorithms such as $WSAT$ [13]. Random walk consists in making from time to time a random movement that is no more guided by the evaluation function. RW constitutes a kind of diversification. Integrating RW into the basic TS algorithm gives us a diversifying $TS+RW$ algorithm.

At each iteration of the $TS+RW$ algorithm, a real $rw \in [0, 1]$ is randomly chosen. Let $q \in [0, 1]$ be a given threshold value, if $rw > q$, then the algorithm does a classical movement (see section 2), else the algorithm does a random and feasible movement. So, the $TS+RW$ algorithm may be described as follows:

In: a feasible configuration x_{init} , the number of iterations L , the random threshold q
Out: a new feasible configuration x

```

 $x \leftarrow x_{init}$ 
for  $i = 0$  to  $L$  do
  generate a random value  $rw \in [0, 1]$ 
  if random value  $rw < q$  then
    choose a random authorized move  $m$ 
  else
    choose the best authorized move  $m$ 
  end if
  update the tabu list with  $m$ 
  perform the chosen move  $m$  in  $x$ 
  update the set of non-dominate configurations with  $x$ 
end for

```

Algorithm 2: The random-walk Tabu algorithm $TS+RW$.

So setting q to 0 leads to the basic TS algorithm while setting q to 1 gives us a random search algorithm.

The Hamming distance based tabu algorithm $TS+HW$. Our third and last Tabu algorithm is based on an original and more rational diversification technique (with regard to $TS+RW$). The basic idea consists in supervising the evolution of the diversity of some recently visited configurations. If the diversity drops below some threshold, a diversification phase is executed.

The diversity of the configurations is calculated using *Hamming Distance*. To apply the basic idea, one needs a way of calculating efficiently and incrementally the hamming distance for the given configurations. The dedicated formula and an incremental technique are described at ap-

pendix A.

So at each iteration of the $TS+HW$ algorithm, the diversity of the l last visited configurations (l is fixed empirically) is calculated. If the value of the diversity is lower than a given threshold value d , then the algorithm changes its behavior and starts a diversification phase. Different ways are possible for the diversification phase. In this paper, we just increase the Tabu tenure of some highly repeated moves. Then, the search restarts from a deteriorated configuration (for example, the configuration whose variables are set to 0). The $TS+HW$ algorithm is then described as follows :

In: a feasible configuration x_{init} , the number of iterations L , the diversity threshold d
Out: a new feasible configuration x

```

 $x \leftarrow x_{init}$ 
for  $i = 0$  to  $L$  do
  if diversity level  $< d$  then
    increase tabu tenure for the most frequent moves done during the last intensification phase
     $x \leftarrow$  configuration-zero
  end if
  choose the best authorized move  $m$ 
  update the tabu list with  $m$ 
  perform the chosen move  $m$  in  $x$ 
  update the set of non-dominate configurations with  $x$ 
end for

```

Algorithm 3: The Hamming distance based Tabu algorithm $TS+HW$.

3 Experimental study of the role of diversity

In this section, we carried out experimental studies of the three tabu algorithms to highlight the importance of diversification in the context of multi-criteria optimization.

Test data

The experiments are based on three of the nine instances that were used in [20]. These instances have 2 objectives, with respectively 250, 500 and 750 items. Moreover, for each instance, the number of constraints is equal to 2. The instances were generated randomly with uncorrelated profits and weights, and the capacities of the knapsack constraints were set to be half of the total weight regarding the corresponding constraint. As a result, half of the items are expected to be in the optimal solutions.

Performance measures

In order to evaluate the results (the trade-off front) produced by the different algorithms, we use two measures which are scaling-independent with regard to each objective criterion: *The size of the dominated space (S)* and *the coverage of two sets (C)* [19].

Instance		Method	
Number of objectives	Number of items	Tabu List size (all algorithms)	Number of iterations (all algorithms)
2	250	19	100 000
	500	21	200 000
	750	23	400 000

Table 1: Parameters settings for different algorithms and instances.

Definition 1 The size of the dominated space (\mathcal{S}). Let $A = (x_1, x_2, \dots, x_l) \subseteq X$ be a set of l decision vectors. The function \mathcal{S} gives the volume enclosed by the union of the polytopes p_1, p_2, \dots, p_l , where each p_i is formed by the intersections of the following hyper-planes arising out of x_i , along with the axes: for each axis in the objective space, there exists a hyper-plane perpendicular to the axis and passing through the point $(f_1(x_i), f_2(x_i), \dots, f_k(x_i))$. In the two-dimensional case, each p_i represents a rectangle defined by the points $(0, 0)$ and $(f_1(x_i), f_2(x_i))$.

This measure cannot be used to compare two sets relatively to each other. In order to determine the dominance ratio between two sets, we apply a second measure.

Definition 2 The coverage of two sets (\mathcal{C}). Let $A, B \subseteq X$ be two sets of decision vectors. The function \mathcal{C} maps the ordered pair (A, B) to the interval $[0, 1]$:

$$\mathcal{C}(A, B) := \frac{|\{b \in B \mid \exists a \in A : a \geq b\}|}{|B|}$$

The value $\mathcal{C}(A, B) = 1$ means that all decision vectors in B are weakly dominated by A . The opposite, $\mathcal{C}(A, B) = 0$, represents the situation when none of the points in B are weakly dominated by A .

Experimental settings

The TS algorithms are programmed in CAML and compiled using OCAML. In order to get fair comparisons, the most important data structures are shared by the three studied algorithms.

In our experimentation, the following empirical settings are used:

- for $TS+RW$, the threshold value for random walk q is set to 0.15 ;
- for $TS+HW$, the threshold value for diversity level value d is set to 0.15.

Table 1 summarizes the settings of the other main parameters used by TS , $TS+RW$ and $TS+HW$. Table 2 shows, for each of the three compared algorithms and for each problem instance, the computing time obtained from the above parameter settings². Each problem instance is solved ten times with each algorithm.

²Timing is based on binary codes generated by the compiler OCAML and a PC running Linux (Bi-Pentium III 1 Ghz).

Number of Objectives	Number of Items	TS	$TS+RW$	$TS+HW$
2	250	27	24.86	37.96
	500	113.32	105.58	151.88
	750	336.26	304.96	456.67

Table 2: Average running times given to each algorithm (seconds).

Number of Objectives	Number of Items	TS	$TS+RW$	$TS+HW$
2	250	8.97e+7	9.02e+7	9.84e+7
	500	3.72e+8	3.74e+8	4.06e+8
	750	7.96e+8	8.04e+8	8.88e+8

Table 3: Average of the size of the dominated space \mathcal{S} .

Comparisons

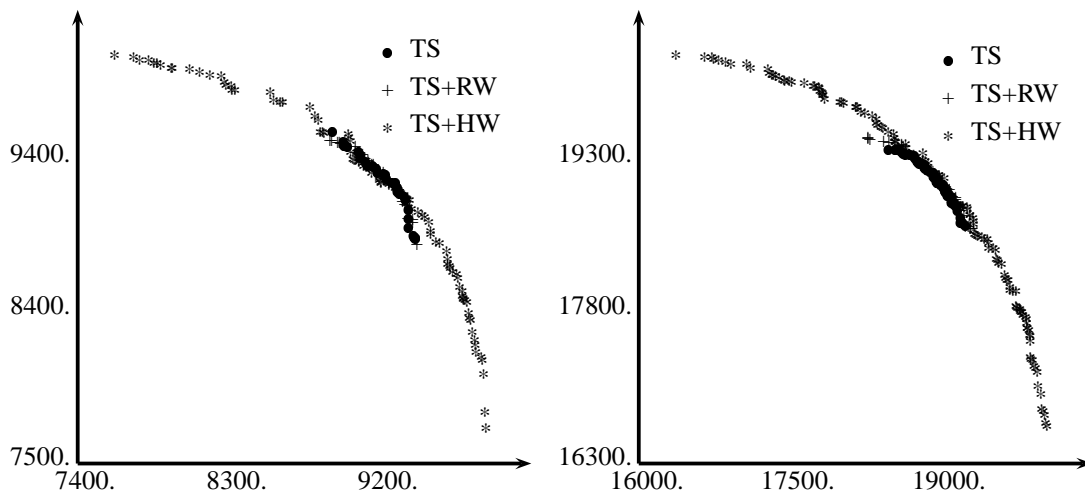
In this subsection, we present experimental results of the three Tabu search algorithms. Results shown below for each instance represent average ones from ten independent runs.

Table 3 shows the results for the \mathcal{S} measure (size of the dominated space). From the table, one observes first that both $TS+HW$ and $TS+RW$ give significantly better results (larger \mathcal{S} values) than TS for all the instances. By comparing $TS+HW$ and $TS+RW$, one observes that $TS+HW$ outperforms $TS+RW$ for all the instances with a better gap than between TS and $TS+RW$. One may also notice the remarkable results of the $TS+HW$ on the two largest instances.

Following [20], figure 1 summarizes the results on the \mathcal{C} measure (set coverage). On this figure, each small black bar represents the results of the \mathcal{C} measure between two methods, for each problem instance. Two values are represented: the gap between the max and min values encountered during the ten runs, and the average of the \mathcal{C} measure on all the runs.

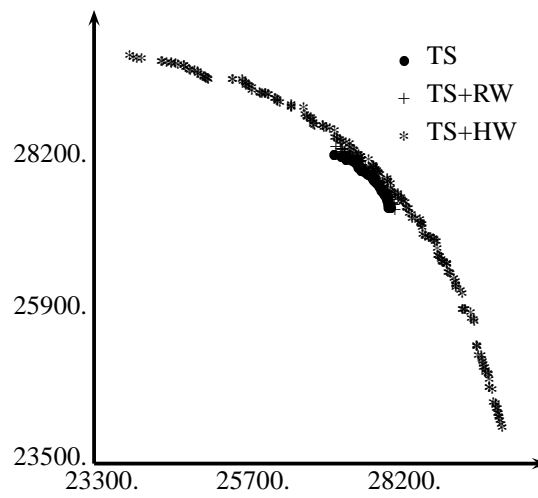
It is observed that once again the results of TS are inferior to those of $TS+RW$ and $TS+HW$ for all the instances. By comparing $TS+RW$ and $TS+HW$, we observe that the results are in favor of $TS+HW$ for all the instances. Indeed, the values obtained by $\mathcal{C}(TS+RW, TS+HW)$ are smaller than the values obtained by $\mathcal{C}(TS+HW, TS+RW)$. We notice that on the hardest instances (500 and 750 items) the factor rate between the results is greater than 10.

Figure 2 shows the set of non dominated points discovered by each algorithm. We can notice that the performance hierarchy observed above is respected here. Thus, TS curve is the more restricted on the three instances. The curve of $TS+RW$ method is at the second place, it is a little bit more extended than TS one. The curve obtained by $TS+HW$ is clearly much better than that of TS and $TS+RW$. Some areas, where the curves are superimposed, are more difficult to study. In these areas, the algorithms get some comparable results. Finally, it is noticed that the more the instance becomes large and hard, the more the behavior of $TS+HW$ is good compared with TS and $TS+RW$.



Instance with 250 items

Instance with 500 items



Instance with 750 items

Figure 2: Non dominated points of the objective space.

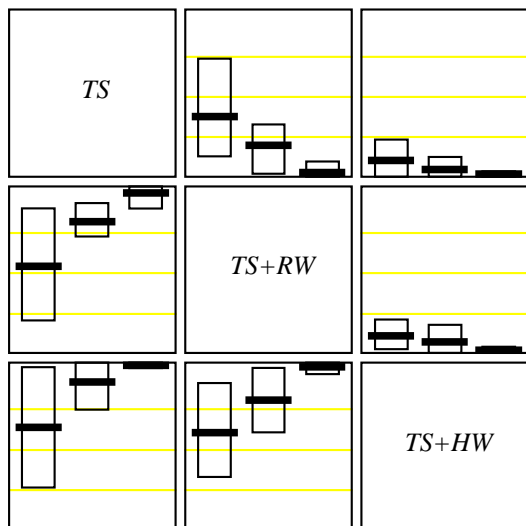


Figure 1: Results of the comparison of TS , $TS+RW$ and $TS+HW$ with \mathcal{C} measure. Each chart contains three box plots representing the distribution of \mathcal{C} values for a certain ordered pair of algorithms; the three box plots from left to right relate to 250, 500 and 750 items. The scale is 0 at the bottom and 1 at the top of each chart. Furthermore, each rectangle refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives the fraction of B weakly dominated by A : $\mathcal{C}(A, B)$.

Discussions

The results show that the $TS+HW$ algorithm performs better than $TS+RW$ and TS . When we increase the number of iterations for $TS+RW$ and TS , the quality of the results improves but the area covered by both algorithms is always smaller than that covered by $TS+HW$. The $TS+HW$ algorithm explores better the objectives space.

Therefore, diversification based on tabu list alone or combined with random walk is not sufficient for an efficient search. On the contrary, the diversification based on hamming distance and used by $TS+HW$ is a much more effective mechanism. This mechanism allows the algorithm to diversify the search when it is needed and helps to adjust the tabu tenure in a dynamic and adaptive way.

4 Comparisons with other well-known algorithms

In this section, we compare the best Tabu algorithm, *i.e.* $TS+HW$, with $SPEA$ [20] and $MOGLS$ [9], two well-known evolutionary algorithms for the MOKP. To our knowledge, $MOGLS$ is the algorithm that gets the best results on the tested instances (see [8]).

Number of Objectives	Number of Items	$SPEA$	$MOGLS$	$TS+HW$
2	250	7.48	23.41	18.50
	500	25.66	48.64	43.95
	750	56.16	82.33	71.87

Table 5: Average running times given to each algorithm (seconds).

Number of Objectives	Number of Items	$SPEA$	$MOGLS$	$TS+HW$
2	250	9.40e+7	9.86e+7	9.84e+7
	500	3.82e+8	4.07e+8	4.05e+8
	750	8.06e+8	8.92e+8	8.77e+8

Table 6: Average of the size of the dominated space \mathcal{S} .

Experimental settings

Like $TS+HW$, the $SPEA$ and $MOGLS$ algorithms are programmed in CAML and compiled using OCAML. Once again, the most important data structures are shared by the three algorithms. This provides us a solid basis for a fair comparison between these algorithms.

In our experimentation, the following settings are used:

- for $TS+HW$, the threshold value for diversity threshold d is set to 0.15;
- for $SPEA$, we set the mutation rate to 0.2, and the clustering level to 15000, according to [20] and [8];
- the population sizes for $SPEA$ and $MOGLS$ were set according to the number of items and the number of objectives of the instance, but the same size is used for the three algorithms for a given instance (see table 4).

Table 4 summarizes the settings of the main parameters used by TS , $SPEA$ and $MOGLS$. Table 5 shows, for each of the three compared algorithms and for each problem instance, the computing time obtained from the above parameter settings. Each problem instance is solved ten times with each algorithm.

Comparisons of $TS+HW$, $SPEA$ and $MOGLS$

Table 6 shows the averaged results for the \mathcal{S} measure (size of the dominated space). From the table, one observes first that both $TS+HW$ and $MOGLS$ give significantly better results (larger \mathcal{S} values) than $SPEA$ for all the instances. By comparing $TS+HW$ and $MOGLS$, one observes that $MOGLS$ outperforms slightly $TS+HW$ for all the instances with a smaller gap than between $SPEA$ and $TS+HW$.

As seen at previous section 3, figure 3 summarizes the results on the \mathcal{C} measure (set coverage). It is observed that the results of $MOGLS$ are superior to those of $SPEA$ and $TS+HW$ for all the instances. By comparing $SPEA$ and $TS+HW$, we observe that $TS+HW$ is as good as $SPEA$ for the first instance. For the two last instances, the results are clearly in favor of $TS+HW$. Indeed, the values obtained by

Instance		Method					
Number of objectives	Number of items	Initial population size		Number of generation		Number of iterations	Tabu list size
		<i>SPEA</i>	<i>MOGLS</i>	<i>SPEA</i>	<i>MOGLS</i>	<i>TS+HW</i>	<i>TS+HW</i>
2	250	150	150	50	50	50000	19
	500	200	200	50	50	55000	21
	750	250	250	50	50	60000	23

Table 4: Parameters settings for different algorithms and instances.

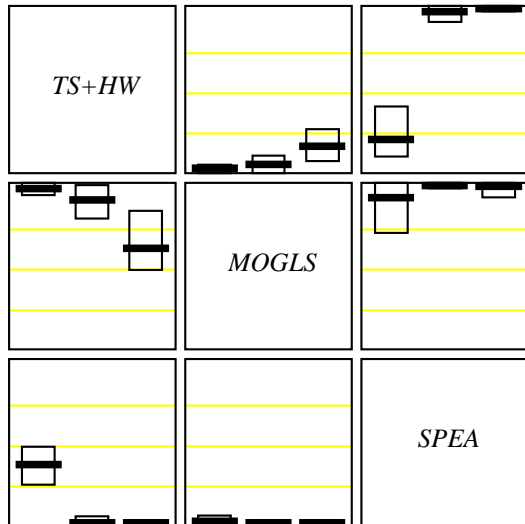


Figure 3: Results of the comparison of *TS+HW*, *MOGLS* and *SPEA* with \mathcal{C} measure. Each chart contains three box plots representing the distribution of \mathcal{C} values for a certain ordered pair of algorithms; the three box plots from left to right relate to 250, 500 and 750 items. The scale is 0 at the bottom and 1 at the top of each chart. Furthermore, each rectangle refers to algorithm A associated with the corresponding row and algorithm B associated with the corresponding column and gives the fraction of B weakly dominated by A : $\mathcal{C}(A, B)$.

$\mathcal{C}(SPEA, TS+HW)$ are smaller than the values obtained by $\mathcal{C}(TS+HW, SPEA)$. We notice that on the hardest instances (500 items, and 750 items) the results of *TS+HW* widely exceed those of *SPEA*.

Figure 4 represents the set of non dominated points discovered by each algorithm. It should be noticed that on the 250 and 350 items instances, *MOGLS* gets the best curves and confirms the results observed with the other measures. However, for the largest instance (750 items), *TS+HW* dominates some points found by *MOGLS* in some sections of its non dominated set. This is remarkable given that *TS+HW* operates with a single configuration while *MOGLS* (as well as *SPEA*) uses a population of 250 configurations.

TS+HW seems to perform better with regard to the other algorithms on larger instances. In fact, for such an instance, *TS+HW* is able to discover solutions which are not dominated by the other methods. It also finds solutions which dominate some solutions found by *MOGLS* and *SPEA*. In other words, *TS+HW* and *MOGLS* (or *SPEA*) discovers dif-

ferent and non dominated points because they explore in quite different ways the search space and the objective space. This observation gives a justification for the genetic local search approach which combines genetic and local search into a single algorithm [9].

5 Conclusion

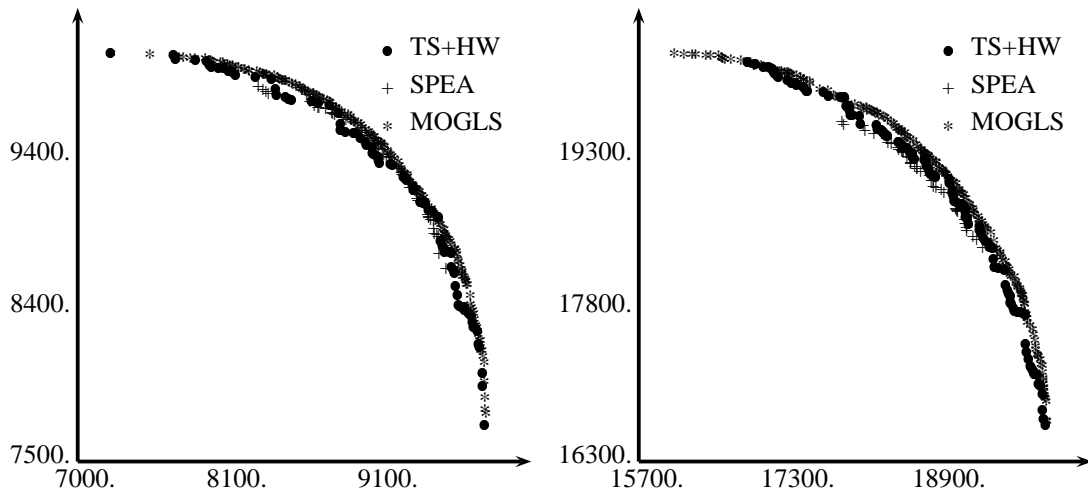
In this paper, we have presented an empirical study of Tabu Search for the 0-1 multidimensional multi-objective knapsack problem (MOKP). We have studied three TS algorithms *TS*, *TS+RW* and *TS+HW* which are based on different diversification techniques. Experimental results show that the *TS+HW* algorithm, which integrates a Hamming distance based diversification mechanism, proved to be quite efficient.

Experiments were also carried out to assess the performance of *TS+HW* with respect to two leading hybrid evolutionary algorithms *SPEA* and *MOGLS*. Results showed that even if the TS algorithm operates with a single solution during its search, it outperforms *SPEA* and competes favorably in some cases with *MOGLS*. This is remarkable given the simplicity of *TS+HW* compared with these population based algorithms. Indeed, both *SPEA* and *MOGLS* are based on the more sophisticated evolutionary models and integrate local search within them.

The *TS+HW* algorithm may be improved in several ways. First, based on the diversity monitoring technique, more elaborated techniques may be used for the diversification phase. Second, constraint handling techniques developed for the classical 0-1 multidimensional multi-objective knapsack problem [17] could be very useful in the context of the MOKP.

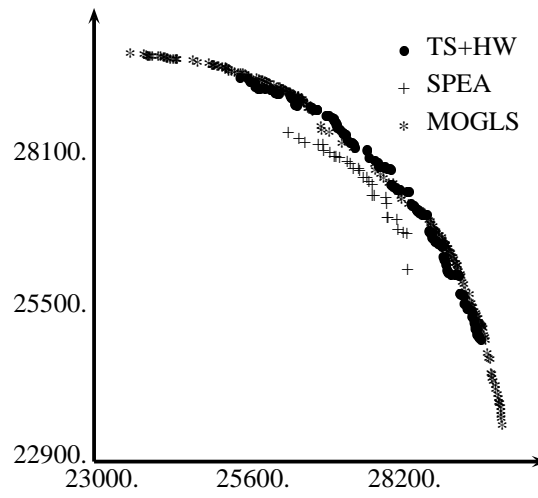
This study highlights the importance of an effective management of diversity during a search process for multi-objective optimization and this remains true for both evolutionary search and local search. This study also suggests the possibility of integrating TS within the evolutionary framework, leading to even more powerful hybrid algorithms for multi-objective optimization.

Acknowledgements: This work is partially supported by two grants from the LIAMA and the PRA program (SI00-01) which are greatly acknowledged.



Instance with 250 items

Instance with 500 items



Instance with 750 items

Figure 4: Non dominated points of the objective space.

References

- [1] J.L. Cohon. Multiobjective programming and planning. *Mathematics in Science and Engineering, Academic Press, Inc*, 140, 2000.
- [2] D.W. Corne and J.D. Knowles. M-paes : a memetic algorithm for multiobjective optimization. In *Proceedings of the 2000 congress on evolutionary computation*, pages 325–332, 2000.
- [3] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley, 2001.
- [4] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, (22):425–460, 2000.
- [5] X. Gandibleux, N. Mezdaoui, and A. Freville. A multiobjective tabu search procedure to solve combinatorial optimization problems. In *Lecture Notes in Economics and Mathematical Systems*, volume 455, pages 291–300. Springer, 1997.
- [6] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [7] D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. *Reading, MA: Addison-Wesley*, 1989.
- [8] A. Jaskiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem : a comparative experiment. Technical Report RA-002, Research report, Institute of Computing Science, Poznan University of Technology, 2000.
- [9] A. Jaskiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.
- [10] K. Miettinen. *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, 1999.
- [11] W. Morrison and K.A De Jong. Measurement of population diversity. In *Artificial Evolution*, pages 31–41. Springer, 2001.
- [12] J.D. Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University, 1984.
- [13] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *AAAI*, volume 1, pages 337–343, 1994.
- [14] P. Serafini. Simulated annealing for multiobjective optimization problems. In *10th Int. Conf. on MCDM*, pages 87–96, 1992.
- [15] N. Srinivas and K. Deb. Multiobjective optimization using non dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
- [16] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. Mosa method : a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, (8):221–336, 1999.
- [17] M. Vasquez and J.K. Hao. A hybrid approach for the 0-1 multidimensional knapsack problem. In *Proc. of the 13th Intl. Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 328–333, 2001.
- [18] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization : the strength pareto approach. Technical Report 43, 1998.
- [19] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms : a comparative case study. In *Lecture Notes in Computer Science*, pages 292–301. Springer, 1998.
- [20] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, (3):257–271, 1999.

A Hamming distance and its incremental evaluation.

The Hamming distance between two configurations $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ is defined as follows:

$$\sum_{i=1}^n |x_i - y_i|, \quad x_i, y_i \in \{0, 1\}$$

More generally, the sum of the Hamming distances SH of a set of p configurations with n variables is defined by:

$$SH = \sum_{j=1}^{p-1} \sum_{j'=j+1}^p \sum_{i=1}^n |y_{ij} - y_{ij'}|, \quad y_{ij}, y_{ij'} \in \{0, 1\}$$

So the value SH represents the sum of the Hamming distances of the p configurations that are taken two by two. This measurement is computed in a quadratic time with regard to p . So it is not easily usable when p is huge or when a very huge number of measurements must be done.

Recently, R.W Morrison and K.A De Jong [11] have proposed a transformation of the classical formula. The new transformed formula is now computable in a linear time. It is defined by:

$$\text{Let } w_i = \frac{\sum_{t=1}^p x_{it}}{p}$$

$$\text{then } SH = p \times \left(\sum_{i=1}^n \sum_{j=1}^p (x_{ij} - w_i)^2 \right)$$

This linear formula is not in an incremental form. By substituting w_i by its expression, and by developing the formula, we obtain the reasoning given at figure 5.

Under this form, SH becomes incremental and so can be efficiently used in an incremental process.

$$\begin{aligned}
SH &= p \times \left(\sum_{i=1}^n \sum_{j=1}^p \left(x_{ij} - \frac{\sum_{t=1}^p x_{it}}{p} \right)^2 \right) \\
SH &= p \times \left(\sum_{i=1}^n \sum_{j=1}^p \left(x_{ij}^2 - \frac{2}{p} \times x_{ij} \times \sum_{t=1}^p x_{it} + \left(\frac{\sum_{t=1}^p x_{it}}{p} \right)^2 \right) \right) \\
\text{let } SH &= p \times \sum_{i=1}^n \left(\sum_{j=1}^p x_{ij}^2 - \frac{2}{p} \times \left(\sum_{t=1}^p x_{it} \right) \times \left(\sum_{j=1}^p x_{ij} \right) + \sum_{j=1}^p \left(\frac{\sum_{t=1}^p x_{it}}{p} \right)^2 \right) \\
\text{with changes of variable:} \\
SH &= p \times \sum_{i=1}^n \left(\sum_{j=1}^p x_{ij}^2 - \frac{2}{p} \times \left(\sum_{j=1}^p x_{ij} \right)^2 + \frac{p}{p^2} \times \left(\sum_{j=1}^p x_{ij} \right)^2 \right) \\
\text{as } x_{ij} \in \{0, 1\} \text{ we have } x_{ij}^2 &= x_{ij} \\
\text{and } SH &= p \times \sum_{i=1}^n \left(\sum_{j=1}^p x_{ij} - \frac{1}{p} \times \left(\sum_{j=1}^p x_{ij} \right)^2 \right) \\
\text{let } SH &= \sum_{i=1}^n \left(\left(\sum_{j=1}^p x_{ij} \right) \times \left(p - \sum_{j=1}^p x_{ij} \right) \right)
\end{aligned}$$

Figure 5: Transformation of the R.W Morrison and K.A De Jong formula in an incremental form.