Modified Choice Function Heuristic Selection for the Multidimensional Knapsack Problem

John H. Drake[1], Ender Özcan[1] and Edmund K. Burke[2]

[1] ASAP Research Group
School of Computer Science, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{psxjd2,ender.ozcan}@nottingham.ac.uk
[2] Computing Science and Mathematics, School of Natural Sciences
University of Stirling, Stirling, FK9 4LA, Scotland
e.k.burke@stir.ac.uk

Abstract. Hyper-heuristics are a class of high-level search methods used to solve computationally difficult problems, which operate on a search space of low-level heuristics rather than solutions directly. Previous work has shown that selection hyper-heuristics are able to solve many combinatorial optimisation problems, including the multidimensional 0-1 knapsack problem (MKP). The traditional framework for iterative selection hyper-heuristics relies on two key components, a heuristic selection method and a move acceptance criterion. Existing work has shown that a hyper-heuristic using *Modified Choice Function* heuristic selection can be effective at solving problems in multiple problem domains. *Late Acceptance Strategy* is a hill climbing metaheuristic strategy often used as a move acceptance criteria in selection hyper-heuristics. This work compares a *Modified Choice Function - Late Acceptance Strategy* hyper-heuristic to an existing selection hyper-heuristic method from the literature which has previously performed well on standard MKP benchmarks.

Keywords: Hyper-heuristics, Choice Function, Heuristic Selection, Multidimensional Knapsack Problem, Combinatorial Optimization

1 Introduction

Hyper-heuristics are high-level search methodologies which operate on a search space of heuristics. A hyper-heuristic is defined by Burke et al. [1, 2] as: '...a search method or learning mechanism for selecting or generating heuristics to solve computational search problems'. This definition covers the two main classes of hyper-heuristics, those concerned with heuristic selection and those with heuristic generation. Although often considered as an alternative to metaheuristics, the recent definition of a metaheuristic offered by Sörensen and Glover [3] somewhat subsumes hyper-heuristics. According to their definition, a selection hyper-heuristic is a metaheuristic which provides a framework within which to mix and control low-level heuristics, whereas a generation hyper-heuristic is a metaheuristic to generate heuristics. It follows that if a metaheuristic is a heuristic, a hyper-heuristic can either be a metaheuristic itself (e.g. a Grammatical Evolution system to generate heuristics [4]) or contain metaheuristic components (e.g. a selection hyper-heuristic using *Late Acceptance Strategy* move acceptance criterion [5]). Hyper-heuristics have been applied successfully to a wide range of problems including: production scheduling [6], nurse rostering [7], examination timetabling [7], sports scheduling [8], bin packing [9], dynamic environments [10], vehicle routing [4] and the multidimensional 0-1 knapsack problem [11, 12].

A traditional selection hyper-heuristic iteratively selects and applies low-level heuristics to a single solution, with a decision made at each step whether to accept the new solution. In this paper, such hyper-heuristics are labelled *heuristic selection method - move acceptance criterion* hereafter. Özcan et al. [13] described four different hyper-heuristic frameworks. One of these frameworks F_C, selects and applies a mutational heuristic from a set of low-level heuristics, followed by a pre-defined hill climber before deciding whether to accept the new solution. This is the framework used in this paper, illustrated in Figure 1.

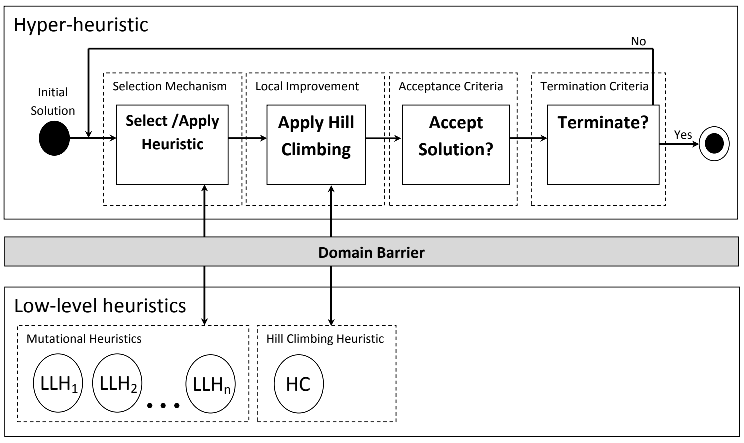

Fig. 1: F_C single-point search hyper-heuristic framework

The *Modified Choice Function* is an elegant heuristic selection method inspired by Reinforcement Learning, which scores heuristics based on a combination of three different measures. At each given step of a search, the heuristic selected is based on a weighted combination of these scores. Existing work has shown that a hyper-heuristic using *Modified Choice Function* heuristic selection can be effective at solving problems in multiple problem domains. In this paper we investigate the suitability of using *Modified Choice Function* hyper-heuristics to solve the MKP, a domain in which hyper-heuristics have previously shown to be successful.

2 Selection Hyper-heuristics and *Choice Function* Heuristic Selection

The first use of the term hyper-heuristic was by Cowling et al. [14], who defined hyper-heuristics as '*heuristics to choose heuristics*'. This paper investigated the application of a number of *Simple Random*, *Greedy* and *Choice Function*-based hyper-heuristic approaches to a real-world sales summit scheduling problem using two deterministic move acceptance criteria: *All Moves* and *Only Improving. Choice Function* heuristic selection has also been used by Bilgin et al. [15] for benchmark function optimisation, Özcan et al. [16] and Burke et al. [17] for examination timetabling and Drake et al. [12] for the MKP.

The *Choice Function* is a heuristic selection method which scores heuristics based on a combination of three different measures. The first measure (f_1) reflects the previous performance of a given low-level heuristic, weighted towards the most recent application. The value of f_1 for a low-level heuristic h_j is defined as:

$$f_1(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \tag{1}$$

where $I_n(h_j)$ is the change in solution quality, $T_n(h_j)$ is the time taken to execute the heuristic for each previous invocation n of heuristic h_j and α is a value between 0 and 1 giving greater importance to recent performance.

The second measure (f_2) attempts to capture any pair-wise dependencies between heuristics. Values of f_2 are calculated for each heuristic h_j when invoked immediately following h_k using the formula in Equation 6:

$$f_2(h_k, h_j) = \sum_n \beta^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} \tag{2}$$

where $I_n(h_k, h_j)$ is the change in evaluation function, $T_n(h_k, h_j)$ is the time taken to call the heuristic for each previous invocation n of heuristic h_j following h_k and β is a value between 0 and 1 which also gives greater importance to recent performance.

The third measure (f_3) is the time elapsed ($\tau(h_j)$) since the heuristic was last selected by the *Choice Function.* This allows all heuristics at least a small chance of selection.

$$f_3(h_j) = \tau(h_j) \tag{3}$$

In order to rank heuristics, a score is given to each heuristic with *Choice Function F* calculated as:

$$F(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \delta f_3(h_j) \tag{4}$$

where α and β as defined previously weight f_1 and f_2 respectively to provide intensification of the heuristic search process whilst δ weights f_3 to provide sufficient diversification.

2.1 *Modified Choice Function* Heuristic Selection

The *Modified Choice Function* [18] was introduced to overcome some of the limitations of the classic *Choice Function*, when applied to cross-domain optimisation using the CHeSC2011 [19] benchmarks. *Modified Choice Function* heuristic selection automatically controls the intensification and diversification parameters of the *Choice Function*, weighting each of f_1, f_2 and f_3 using a method inspired by Reinforcement Learning, giving far greater emphasis to intensification. The *Modified Choice Function* does not make a distinction between the values of α or β which weight f_1 and f_2, so these are reduced to a single intensification parameter which is referred to as ϕ. Using the *Modified Choice Function*, the score F_t for each heuristic h_j is defined as:

$$F_t(h_j) = \phi_t f_1(h_j) + \phi_t f_2(h_k, h_j) + \delta_t f_3(h_j) \tag{5}$$

where t is the current iteration. At each stage if an improvement in solution quality is observed ϕ is rewarded heavily whilst δ is harshly punished. If the quality of the solution does not improve following the application of a low-level heuristic, the level of intensification is decreased linearly, reducing ϕ and increasing δ to diversify the heuristic search process. The intention is to give the intensification component of the *Choice Function* more time as the dominating factor in the calculation of F. In this paper the parameters ϕ_t and δ_t are defined as:

$$\phi_t = \begin{cases} 0.99, & \text{if an improving move is made} \\ \max\{\phi_{t-1} - 0.01, 0.01\}, & \text{if a non-improving move is made} \end{cases} \tag{6}$$

$$\delta_t = 1 - \phi_t \tag{7}$$

The use of 0.01 as the minimum value ensures that the diversification component of the *Modified Choice Function* always has some non-negative influence on the F value for each heuristic. Here we would like to clarify that whilst each individual heuristic has its own F value, all low low level heuristics use the same ϕ and δ values. The *Modified Choice Function* was shown to outperform the original *Choice Function* on average, over six different problem domains by Drake et al. [18]. Additionally the best results in the literature were achieved for the MAX-SAT problem domain, within the HyFlex framework.

3 The Multidimensional Knapsack Problem

The NP-hard [20] multidimensional 0-1 knapsack problem (MKP) [21] is a generalised case of the standard 0-1 knapsack problem, with roots in applications such as capital budgeting and project selection. The MKP is a resource allocation model, where the objective is to select a subset of objects which yield the greatest profit, whilst observing the constraints on knapsack capacities. Unlike the standard 0-1 knapsack problem, each object j consumes a different amount of resources in each dimension i when selected. Formally the MKP is stated as:

$$\begin{aligned}
&\text{maximise} && \sum_{j=1}^{n} p_j x_j && && (8)\\
&\text{subject to} && \sum_{j=1}^{n} a_{ij} x_j \leq b_i, && i = 1, ..., m && (9)\\
&\text{with} && x_j \in \{0, 1\}, && j = 1, ..., n && (10)
\end{aligned}$$

where p_j is the profit for selecting item j, a_{ij} is the resource consumption of item j in dimension i, b_i is the capacity constraint of each dimension i. Using direct binary encoding, x_1,...,x_n is a set of decision variables indicating whether or not object j is included in the knapsack. The size of a problem is defined by the total number of variables n and the number of dimensions m.

A number of benchmarks sets exist for the MKP, each with different properties[3]. SAC-94 is a benchmark library of MKP instances taken from a number of papers in the literature, often representing real-world examples. These instances are generally small with m ranging from 2 to 30 and n ranging from 10 to 105 with optimal solutions known for all. ORLib is a widely used benchmark library containing 270 instances, split into 27 sets of 10 instances, containing $n \in \{100, 250, 500\}$ variables, $m \in \{5, 10, 30\}$ dimensions and *tightness ratio* $\in \{0.25, 0.50, 0.75\}$. A third benchmark library was introduced by Glover and Kochenbeger [22], referred to here as GK, containing much larger instances of the MKP with n up to 2500 and m up to 100. As optimal solutions are not known for all of the instances in ORLib and GK, performance is often measured using the %-gap distance to the solution to the LP-relaxed problem calculated as:

$$100 * \frac{LPopt - SolutionFound}{LPopt} \tag{11}$$

where $LPopt$ is the fitness value of the LP-relaxed solution to a given problem and $SolutionFound$ is the fitness value of the solution found. In the case of the SAC-94 instances, as optimal solutions are known, performance is measured by the proportion of instances for which the optimal solution is found.

4 Experimental Framework and Parameters

The *Modified Choice Function* described in the previous section is paired with *Late Acceptance Strategy* acceptance criterion and applied to the MKP benchmarks. *Late Acceptance Strategy* [23] is a general purpose optimisation technique which has previously been used as an acceptance criterion in a number of selection hyper-heuristics [16, 11, 5]. A *Choice Function - Late Acceptance Strategy*

[3] All three benchmark instance sets have been standardised and are available in a unified format at: `http://www.cs.nott.ac.uk/~jqd/mkp/index.html`

hyper-heuristic was shown to be the best of nine selection hyper-heuristics for the MKP by Drake et al. [11]. The same framework is used here, containing seven low-level heuristics to select from. A single hill climbing heuristic to be applied after each low-level, as required by an F_C hyper-heuristic framework, is also included. In each case a run for a single instance terminates once 10^6 fitness evaluations have been performed. This allows for direct comparison to the results of the original *Choice Function - Late Acceptance Strategy* hyper-heuristic and a wide range of existing methods in the literature. A single run of each hyper-heuristic is performed on each of the instances in the ORLib and SAC-94 benchmark sets. In the case of the larger GK instances, results are given as the average of 10 runs for each instance.

5 Computational Results

Table 1 shows the results of the *Modified Choice Function - Late Acceptance Strategy* hyper-heuristic over the 270 ORLib instances in terms of average %-gap, along with the results for *Choice Function - Late Acceptance Strategy.* Standard deviations are given as subscript. The results for both hyper-heuristics are very similar over the ORLib instances, with *Choice Function - Late Acceptance Strategy* obtaining an average %-gap of 0.70 and *Modified Choice Function - Late Acceptance Strategy* a %-gap of 0.73, with little difference in standard deviation. An independent Student's t-test within a 95% confidence interval shows no statistically significant difference between the two hyper-heuristics on this benchmark set.

Table 2(a) and Table 2(b) compare the performance of *Choice Function - Late Acceptance Strategy* and *Modified Choice Function - Late Acceptance Strategy* over the SAC-94 and GK problem instances. In the case of the SAC-94 instances, which contains six subsets of problems, the success rate is defined as the proportion on instances within each subset that the optimal solution is found. Again the two methods are showing very similar performance on both of these benchmark sets. *Modified Choice Function - Late Acceptance Strategy* is slightly outperformed in terms of success rate in the pb and weish instances, finding the optimal solution in one and three less instances respectively compared to *Choice Function - Late Acceptance Strategy.* On the GK benchmark set, both methods obtain the same average %-gap over 10 runs of each of the 11 instances.

Table 3 gives the average %-gap for a number of techniques from the literature over the ORLib benchmark set. Our approach is able to outperform many of the existing methods, achieving an average %-gap of 0.73. This is considerably lower than many of the heuristic methods an some of the metaheuristic techniques proposed previously.

6 Conclusions

In this work we have applied a *Modified Choice Function - Late Acceptance Strategy* selection hyper-heuristic to a well known optimisation problem, the

Table 1: Comparison between *Choice Function - Late Acceptance Strategy* (CF-LAS) and *Modified Choice Function - Late Acceptance Strategy* (MCF-LAS) on all 270 instances of ORLib in terms of %-gap

Instance Set	CF-LAS	MCF-LAS
OR5x100-0.25	$1.16_{0.20}$	$1.09_{0.21}$
OR5x100-0.50	$0.53_{0.08}$	$0.57_{0.08}$
OR5x100-0.75	$0.40_{0.07}$	$0.38_{0.05}$
OR5x250-0.25	$0.42_{0.04}$	$0.41_{0.10}$
OR5x250-0.50	$0.20_{0.03}$	$0.22_{0.04}$
OR5x250-0.75	$0.13_{0.01}$	$0.14_{0.02}$
OR5x500-0.25	$0.19_{0.03}$	$0.21_{0.04}$
OR5x500-0.50	$0.10_{0.03}$	$0.10_{0.03}$
OR5x500-0.75	$0.06_{0.01}$	$0.06_{0.01}$
OR10x100-0.25	$2.00_{0.22}$	$1.87_{0.16}$
OR10x100-0.50	$1.02_{0.19}$	$0.95_{0.16}$
OR10x100-0.75	$0.58_{0.08}$	$0.53_{0.09}$
OR10x250-0.25	$0.83_{0.09}$	$0.79_{0.11}$
OR10x250-0.50	$0.39_{0.06}$	$0.41_{0.05}$
OR10x250-0.75	$0.23_{0.03}$	$0.24_{0.03}$
OR10x500-0.25	$0.40_{0.06}$	$0.44_{0.07}$
OR10x500-0.50	$0.18_{0.02}$	$0.20_{0.03}$
OR10x500-0.75	$0.12_{0.01}$	$0.13_{0.01}$
OR30x100-0.25	$3.45_{0.46}$	$3.61_{0.53}$
OR30x100-0.50	$1.56_{0.26}$	$1.60_{0.29}$
OR30x100-0.75	$0.92_{0.08}$	$0.97_{0.15}$
OR30x250-0.25	$1.55_{0.17}$	$1.75_{0.22}$
OR30x250-0.50	$0.71_{0.08}$	$0.79_{0.10}$
OR30x250-0.75	$0.39_{0.04}$	$0.43_{0.07}$
OR30x500-0.25	$0.92_{0.10}$	$1.05_{0.10}$
OR30x500-0.50	$0.39_{0.05}$	$0.44_{0.06}$
OR30x500-0.75	$0.23_{0.02}$	$0.27_{0.02}$
Average_{StdDev}	$0.70_{0.09}$	$0.73_{0.11}$

Table 2: Performance of *Choice Function - Late Acceptance Strategy* and *Modified Choice Function - Late Acceptance Strategy* in terms of (a) success rate of over SAC-94 instances and (b) %-gap obtained in GK instances

(a)

Dataset	CF-LAS	MCF-LAS
hp	0.00	0.00
pb	0.67	0.50
pet	0.50	0.50
sento	1.00	1.00
weing	0.63	0.63
weish	1.00	0.90

(b)

Instance	CF-LAS	MCF-LAS
GK01	$0.57_{1.49}$	$0.58_{1.66}$
GK02	$0.81_{3.86}$	$0.78_{3.75}$
GK03	$0.63_{3.10}$	$0.63_{4.30}$
GK04	$0.91_{3.77}$	$0.86_{5.81}$
GK05	$0.45_{3.00}$	$0.44_{5.83}$
GK06	$0.76_{5.02}$	$0.78_{4.04}$
GK07	$0.19_{6.48}$	$0.22_{2.88}$
GK08	$0.33_{5.68}$	$0.31_{7.60}$
GK09	$0.07_{7.47}$	$0.07_{6.85}$
GK10	$0.14_{8.68}$	$0.14_{11.71}$
GK11	$0.13_{12.34}$	$0.14_{11.10}$
Average	$0.45_{0.30}$	$0.45_{0.29}$

Table 3: Comparison of genetic programming hyper-heuristic to previous approaches over all instances in ORLib in terms of %-gap

Type	Reference	%-gap
MIP	Drake et al. [11] (CPLEX 12.2)	0.52
MA	Chu and Beasley [24]	0.54
Selection HH	Drake et al. [11]	0.70
Selection HH	***Modified Choice Function - Late Acceptance Strategy***	**0.73**
MA	Özcan and Basaran [25]	0.92
Heuristic	Pirkul [26]	1.37
Heuristic	Fréville and Plateau [27]	1.91
Generation HH	Drake et al. [12]	3.04
MIP	Chu and Beasley [24] (CPLEX 4.0)	3.14
Heuristic	Akçay et al. [28]	3.46
Heuristic	Volgenant and Zoon [29]	6.98
Heuristic	Magazine and Oguz [30]	7.69

MKP. Previously, using *Modified Choice Function* heuristic selection was shown to outperform the classic *Choice Function*. Additionally the *Choice Function* has previously worked well with *Late Acceptance Strategy* move acceptance when solving the MKP. Although the *Modified Choice Function* has outperformed the *Choice Function* over multiple domains in the past, this has not been the case when applied to the MKP, with the *Modified Choice Function* offering slightly poorer performance in two of the three benchmark sets tested. Future work will combine the *Modified Choice Function* with other move acceptance criteria, to assess whether there is any variation in performance over the MKP benchmarks. We will also test the *Modified Choice Function* on further benchmark problems in order to better understand the type of problem in which this heuristic selection method can perform well.

References

1. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.: A Classification of Hyper-heuristic Approaches. In: Handbook of Metaheuristics 2nd ed. Springer (2010) 449–468
2. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **64**(12) (2013) 1695–1724
3. Sörensen, K., Glover, F.: Metaheuristics. In: Encyclopedia of Operations Research and Management Science. Springer (2013) 960–970
4. Drake, J.H., Kililis, N., Özcan, E.: Generation of vns components with grammatical evolution for vehicle routing. In: EuroGP 2013. Volume 7831 of LNCS., Springer (2013) 25–36
5. Jackson, W.G., Özcan, E., Drake, J.H.: Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. In: Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI 2013), Surrey, UK, IEEE Press (2013) 228–235
6. Fisher, H., Thompson, G.: Probabilistic learning combinations of local job-shop scheduling rules. In: Factory Scheduling Conference, Carnegie Institute of Technology (1961)
7. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics **9**(6) (2003) 451–470
8. Gibbs, J., Kendall, G., Özcan, E.: Scheduling english football fixtures over the holiday period using hyper-heuristics. In: PPSN 2011. Volume 6238 of LNCS., Springer (2011) 496–505
9. López-Camacho, E., Terashima-Marín, H., Ross, P.: A hyper-heuristic for solving one and two-dimensional bin packing problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011), Dublin, Ireland, ACM (2011) 257–258
10. Kiraz, B., Uyar, A.S., Özcan, E.: Selection hyper-heuristics in dynamic environments. Journal of the Operational Research Society **64**(12) (2013) 1753–1769
11. Drake, J.H., Özcan, E., Burke, E.K.: Controlling crossover in a selection hyper-heuristic framework. Technical Report No. NOTTCS-TR-SUB-1104181638-4244, School of Computer Science, University of Nottingham (2011)

12. Drake, J.H., Hyde, M., Ibrahim, K., Özcan, E.: A genetic programming hyper-heuristic for the multidimensional knapsack problem. In: Proceedings of the 11th IEEE International Conference on Cybernetic Intelligent Systems (CIS 2012), Limerick, Ireland, IEEE Press (2012) 76–80
13. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis **12**(1) (2008) 3–23
14. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: PATAT 2000. Volume 2079 of LNCS., Springer (2001) 176–190
15. Bilgin, B., Özcan, E., Korkmaz, E.E.: An experimental study on hyper-heuristics and exam timetabling. In: PATAT 2006. Volume 3867 of LNCS., Springer (2006) 394–412
16. Özcan, E., Bykov, Y., Birben, M., Burke, E.K.: Examination timetabling using late acceptance hyper-heuristics. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009), Trondheim, Norway, IEEE Press (2009) 997–1004
17. Burke, E.K., Kendall, G., Misir, M., Özcan, E.: Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research **196**(1) (2012) 73–90
18. Drake, J.H., Özcan, E., Burke, E.K.: An improved choice function heuristic selection for cross domain heuristic search. In: PPSN 2012, Part II. Volume 7492 of LNCS., Springer (2012) 307–316
19. Ochoa, G., Hyde, M.: The cross-domain heuristic search challenge (CHeSC 2011). Online (2011) http://www.asap.cs.nott.ac.uk/chesc2011/.
20. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
21. Weingartner, H.M., Ness, D.N.: Methods for the solution of the multidimensional 0/1 knapsack problem. Operations Research **15**(1) (1967) 83–103
22. Glover, F., Kochenberger, G.: Benchmarks for "the multiple knapsack problem". Online (n.d.) http://hces.bus.olemiss.edu/tools.html.
23. Burke, E.K., Bykov, Y.: A late acceptance strategy in hill-climbing for exam timetabling problems. In: Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada (2008) Extended Abstract
24. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics **4**(1) (1998) 63–86
25. Özcan, E., Basaran, C.: A case study of memetic algorithms for constraint optimization. Soft Computing **13**(8-9) (2009) 871–882
26. Pirkul, H.: A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Naval Research Logistics **34**(2) (1987) 161–172
27. Freville, A., Plateau, G.: An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. Discrete Applied Mathematics **49**(1-3) (1994) 189–212
28. Akçay, Y., Li, H., Xu, S.H.: Greedy algorithm for the general multidimensional knapsack problem. Annals of Operations Research **150**(1) (2007) 17–29
29. Volgenant, A., Zoon, J.A.: An improved heuristic for multidimensional 0-1 knapsack problems. Journal of the Operational Research Society **41**(1) (1990) 963–970
30. Magazine, M.J., Oguz, O.: A heuristic algorithm for the multidimensional zero-one knapsack problem. European Journal of Operational Research **16**(3) (1984) 319–326