Computers & Operations Research 43 (2014) 78–89

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

# Generalized quadratic multiple knapsack problem and two solution approaches

Tugba Saraç*, Aydin Sipahioglu

*Industrial Engineering Department of Eskisehir Osmangazi University, Meselik, 26480 Eskisehir Turkey*

ARTICLE INFO

Available online 31 August 2013

*Keywords:*
Generalized Quadratic Multiple Knapsack Problem (G-QMKP)
F-MSG
Genetic Algorithm (GA)
Production with plastic injection
Combinatorial optimization

ABSTRACT

The Quadratic Knapsack Problem (QKP) is one of the well-known combinatorial optimization problems. If more than one knapsack exists, then the problem is called a Quadratic Multiple Knapsack Problem (QMKP). Recently, knapsack problems with setups have been considered in the literature. In these studies, when an item is assigned to a knapsack, its setup cost for the class also has to be accounted for in the knapsack. In this study, the QMKP with setups is generalized taking into account the setup constraint, assignment conditions and the knapsack preferences of the items. The developed model is called Generalized Quadratic Multiple Knapsack Problem (G-QMKP). Since the G-QMKP is an NP-hard problem, two different meta-heuristic solution approaches are offered for solving the G-QMKP. The first is a genetic algorithm (GA), and the second is a hybrid solution approach which combines a feasible value based modified subgradient (F-MSG) algorithm and GA. The performances of the proposed solution approaches are shown by using randomly generated test instances. In addition, a case study is realized in a plastic injection molding manufacturing company. It is shown that the proposed hybrid solution approach can be successfully used for assigning jobs to machines in production with plastic injection, and good solutions can be obtained in a reasonable time for a large scale real-life problem.



## 1. Introduction

The knapsack problem (KP) is a well-known combinatorial optimization problem. The classical KP seeks to select, from a finite set of items, the subset, which maximizes a linear function of the items chosen, subject to a single capacity constraint. In many real life applications, it is important that the profit of packing should also reflect how well the given items fit together. One formulation of such interdependence is the quadratic knapsack problem (QKP). The QKP asks to maximize a quadratic objective function subject to a single capacity constraint. Portfolio management problems, the determination of the optimal sites for communication satellite earth stations or railway stations are good examples of the QKP [8]. The QKP was introduced and solved using a branch-and-bound algorithm by Gallo, Hammer and Simeone in 1980. Later, different branch and bound based solution techniques were offered by Chaillou, Hansen and Mahieu [7], by Michelon and Veuilleux [22] and by Billionnet and Calmels [3]. An exact algorithm was developed by [5] and by Billionnet and Soutif in 2003. In 2005, a greedy Genetic Algorithm was proposed by Julstrom. A survey of upper bounds presented in the literature has been given, and the relative tightness of several of the bounds has been shown by Pisinger [23]. In the same year, Xie and Liu [30] presented a mini-swarm approach for the QKP. In 2009, Sipahioglu and Saraç examined the performance of the modified subgradient algorithm (MSG) to solve the 0-1 QKP and they showed that the MSG algorithm can be successfully used for solving the QKP.

The Quadratic multiple knapsack problem (QMKP) extends the QKP with k knapsacks, each with its own capacity $c_k$. Hiley and Julstrom [13] proposed the first study regarding QMKP in the literature. The paper introduced three heuristic approaches, namely the greedy heuristic, the stochastic hill-climber and the Genetic Algorithm (GA). The greedy heuristic fills the knapsacks one item at a time, always choosing the unassigned item with the highest profit/weight ratio of values to other items with a weight smaller than the remaining capacity of the knapsack. The hill-climber's neighbor operator removes objects from each knapsack, and then refills the knapsack greedily as in the greedy heuristic. The hill-climber's neighbor operator also serves as the GA's mutation. Saraç and Sipahioglu [25] proposed a hybrid genetic algorithm to solve the QMKP. They developed a specialized crossover operator to maintain the feasibility of the chromosomes and presented two distinct mutation operators with different improvement techniques from the non-evolutionary heuristic. They also showed that their GA was more successful than the GA presented by Hiley and Julstrom [13], especially in the case where the number of knapsacks ($k$) increases. In 2007, Singh and Baghel proposed a steady-state grouping genetic algorithm for the QMKP. Like Hiley and Julstrom [13], they also

* Corresponding author. Tel.: +90 2222393750.
*E-mail addresses:* tsarac@ogu.edu.tr, tugba.sarac@gmail.com (T. Saraç), asipahi@ogu.edu.tr (A. Sipahioglu).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.08.018

assumed that all the knapsack capacities $c_k$ where the same. They compared their results with two previously proposed methods; the genetic algorithm and the stochastic hill climber [13]. The results show the effectiveness of their approach. The average solution values obtained were always better than those obtained with the genetic algorithm and the stochastic hill climber. In 2010, [28] proposed a variant of the artificial bee colony algorithm for the QMKP. Their computational results show the superiority of their approach over other approaches in terms of solution quality.

In recent years, studies by McLay and Jacobson [20], Caserta et al. [6], Altay et al. [1], Michel et al. [21] on the knapsack problem that takes setup constraints into consideration have begun to be included in the literature. In these problems, when an item is assigned to a knapsack, the setup cost for its class also has to be attained to the knapsack. Further, not only the weight of the item, but also the cost for the setup has to be taken into consideration in terms of capacity utilization. In the literature, items that required a common setup, rather than separate setups, when being assigned to the same knapsack, were considered as items of the same class or family. McLay and Jacobson [20] provide three dynamic programming algorithms that solve the Bounded Setup Knapsack Problem (BSKP) in pseudo-polynomial time and a fully polynomial-time approximation scheme (FPTAS). One of the dynamic programming algorithms presented solves the Bounded Knapsack Problem (BKP) with the same time and space bounds of the best known dynamic programming algorithm for the BKP. The FPTAS improves the worst-case time bound for obtaining approximate solutions to the BKP compared to using FPTASs designed for the BKP or the 0-1 Knapsack Problem. Caserta et al. [6], proposed a new meta-heuristic based algorithm for the integer knapsack problem with setups. The proposed algorithm is a cross entropy based algorithm, where the meta-heuristic scheme allows a relaxation of the original problem to a series of well-chosen standard knapsack problems, solved through a dynamic programming algorithm. Altay et al. [1], considered a class of knapsack problems that included setup costs for families of items. A mixed integer programming formulation for the problem was provided along with exact and heuristic solution methods. The computational performances of the algorithms were reported and compared with CPLEX. Michel et al. [21] considered multiple-class integer knapsack problem with setups. Their paper provides a review of the literature on knapsack problems with setups, discusses various reformulations, and presents specialized branch-and-bound procedures extending the standard algorithm for the knapsack problem. Sang and Sang [24], proposed a new memetic algorithm for the quadratic multiple container packing problem. The proposed memetic algorithm is based on the adaptive link adjustment evolutionary algorithm (ALA-EA) and it incorporates heuristic fitness improvement schemes into the ALA-EA. Wang et al. [29], provided a comparison of quadratic and linear representations of the QKP based on test problems with multiple knapsack constraints and up to eight hundred variables. In addition to the setup constraints, there may be other important conditions such as the knapsack preference of the items.

In this study, the QMKP with setups is generalized by considering the knapsack preferences of the items and is called the Generalized Quadratic Multiple Knapsack Problem (G-QMKP). It appears to be, it is the first study on generalized quadratic multiple knapsack problems in the literature.

KP that are combinatorial optimization problems belong to the class of NP-hard type problems [18]. Both the QMKP and the G-QMKP are NP-hard by restriction to KP; even if all the quadratic values $p_{ij}$ are set equal to zero and the number of knapsacks equal one. Since solving the G-QMKP is not easy, an efficient search heuristic approach will be useful for solving this problem.

Genetic algorithms are powerful and broadly applicable in stochastic search and optimization techniques based on principles from evolution theory [11]. GAs, which differ from normal optimization and search procedures: (a) work with a coding of the parameter set, not the parameters themselves; (b) search from a population of points, not a single point; (c) use payoff (objective function) information, not derivatives or other auxiliary information; and (d) use probabilistic transition rules, not deterministic rules [12]. Due to these characteristics, use of the GA in numerous fields has been rapidly increasing in the recent years. The GA has also been successfully implemented in the QKP [15] and the QMKP [13,25,26] problems. Another solution method that comes into prominence with its success in solving integer nonlinear problems in recent years [27] is based on the Modified Subgradient (MSG) algorithm. The MSG algorithm was proposed by Gasimov [9] for solving dual problems constructed in respect to a sharp augmented Lagrangean function. If the problem is not convex, using classical Lagrangean based solution methods may lead to a non-zero duality gap. To eliminate this problem, a sharp augmented Lagrangean function is used to construct the dual problem in the MSG algorithm. It is proven that, when the objective and constraint functions are all Lipschitz, then the sharp augmented Lagrangean guarantees the zero duality gap [9]. The MSG algorithm also has some outstanding properties. For example, the MSG algorithm is convergent and does not require any convexity or differentiability conditions on the primal problem. Furthermore, it does not use any penalty parameters, and guarantees the strict increment of dual values at each iteration. Nevertheless, the MSG algorithm also has some disadvantages. It is necessary to find the global optimum of the unconstraint problem at every iteration and uses an upper limit for updating step size parameters. To cope with these problems, the F-MSG algorithm was introduced by [16]. The F-MSG algorithm was used to solve the Quadratic Assignment Problem (QAP) and very good results were obtained [10]. However, the performance of the F-MSG algorithm still depends on the performance of the solution method used for solving the sub problem at any iteration.

In this study, firstly a mathematical model for the G-QMKP is developed and then two different meta-heuristic solution approaches are proposed to solve it. The first method is a genetic algorithm based solution approach and the second method is a hybrid solution approach, which combines the F-MSG (modified subgradient algorithm based on feasible values) and a genetic algorithm for solving the sub problem. Additionally, the success of the proposed methods is shown and compared with the results obtained using the Gams/Dicopt solver on randomly generated test instances. Finally, a case study is realized on a plastic injection molding company as an implementation of the proposed approaches. In this implementation, a large scale real world problem is solved, and it is shown that the proposed hybrid solution approach can be used successfully for assigning jobs to machines in plastic injection production. Moreover, good solutions can be obtained in a reasonable time for a large scale real-life problem.

The organization of this paper is as follows. The second section gives a mathematical model for the G-QMKP. The third section describes the proposed solution methods for both the genetic algorithm and the hybrid algorithm, in detail. Computational results are presented in Section 4. A case study is given in Section 5 and, finally, the conclusions are outlined in Section 6.

## 2. Mathematical model of the generalized quadratic multi knapsack problem

The mathematical model of the Quadratic Multiple Knapsack Problem is as follows:
Index sets:

$J=\{j|j=1,\ldots,n\}$ index set of items
$K=\{k|k=1,\ldots,m\}$ index set of knapsacks

Parameters:

$n$ : the number of items
$m$ : the number of knapsacks
$q_{ij}$ : profit achieved if items i and j are selected for the same knapsack
$p_j$ : profit achieved if item $j$ is selected for any knapsack.
$c_k$ : capacity of knapsack $k$
$w_j$ : the weight of item $j$

Decision variables:

$x_{jk}$:1, if item j is selected for knapsack $k$, 0, otherwise

Model of the QMKP:

$$\max \quad z = \sum_{j=1}^{n}\sum_{k=1}^{m} p_j x_{jk} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{m} q_{ij} x_{ik} x_{jk} \tag{1}$$

*subject to*

$$\sum_{j=1}^{n} w_j x_{jk} \le c_k, \quad \forall k \in K \tag{2}$$

$$\sum_{k=1}^{m} x_{jk} \le 1, \quad \forall j \in J \tag{3}$$

$$x_{jk} \in \{0,1\}, \quad \forall j \in J, \ k \in K \tag{4}$$

where

$$\underset{j}{enb}\ \{w_j\} \le \underset{k}{enk}\ \{c_k\} \quad i, j \in J, \ k \in K$$

The objective (1) of this model is to maximize the total profit. Constraints (2) ensure that the capacity of any knapsack cannot be exceeded. Constraint (3) guarantees that item $j$ can only be selected for one knapsack. Binary restrictions on decision variables are imposed by constraints (4).

In this study, the QMKP is generalized by considering three different constraints: Setup constraint, assignment conditions and knapsack preferences of the items. These constraints are explained in detail as follows:

- *Setup constraint*: In the literature, items that require a common setup, rather than separate setups, when being assigned to the same knapsack, are considered as items of the same class. In this study, each item belongs to a class. To reflect this point of view to the model, the index set, parameters and decision variables related with the setup constraint should be described as below:
  $R=\{r|r=1,\ldots, h\}$ index set of items class
  $s_r$: setup time of the items that belong to class $r$ (if the items belonging to the same class are selected for the same knapsack, only one setup is needed for all).and

$$y_{rk} = \begin{cases} 1 & \text{if any item in class } r \text{ is selected for knapsack } k \\ 0 & \text{otherwise} \end{cases}$$

  In order to consider the setup constraint in the model, constraint (5) has to be changed with constraint (2) in the model.

$$\sum_{j=1}^{n} w_j x_{jk} + \sum_{r=1}^{h} s_r y_{rk} \le c_k, \quad \forall k \in K \tag{5}$$

- *Assignment conditions*: In some cases, all classes may not be selected for every knapsack. In order to consider this special case, a new parameter should be defined and added to the model as given below:

$$t_{rj} = \begin{cases} 1 & \text{if item } j \text{ is in class } r \\ 0 & \text{otherwise} \end{cases}$$

$$\sigma_{rk} = \begin{cases} 1 & \text{if any item in class } r \text{ can be selected for knapsack } k \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{jk} = \begin{cases} 1 & \text{if item } j \text{ can be selected for knapsack } k \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{jk} = \sum_{r=1}^{h} \sigma_{rk} t_{rj}$$

  Assignment constraints where some items and classes cannot be assigned to certain knapsacks are included in the model by formulating the $x_{jk}$ decision variable with the condition $\beta_{jk} > 0$ and the $y_{rk}$ with the condition$\sigma_{rk} > 0$. In this way, the number of decision variables is also reduced.
  Moreover, another assignment constraint for classes can also be considered. In this case, the maximum number of knapsacks which the items in class $r$ can be selected for is limited. Let, $\eta_r$ be the maximum number of knapsacks which the items in class $r$ can be selected for. Then constraint (6) should be added to the model to take this limitation.

$$\sum_{(k|\sigma_{rk}>0)=1}^{m} y_{rk} \le \eta_r \quad \forall r \in R \tag{6}$$

- *Knapsack preferences of the items*: In classical QKP and QMKP, if item $j$ is selected for any knapsack, the profit ($p_j$) is achieved. If the items have knapsack preferences, the profit ($p_{jk}$) may be changed for different knapsacks. To consider the knapsack preferences in the model, the profit parameter should be identified as $p_{jk}$, instead of $p_j$. Let,

  $p_{jk}$: profit achieved if item j is selected for knapsack $k$.

After all the modifications, the developed model is called the Generalized Quadratic Multiple Knapsack Problem (G-QMKP) and the mathematical model of the G-QMKP is given as follows:

Model of the G-QMKP:

$$enb \quad z = \sum_{k=1}^{m}\sum_{(j|\beta_{jk}>0)=1}^{n} p_{jk} x_{jk} + \sum_{k=1}^{m}\sum_{(i|\beta_{ik}>0)=1}^{n-1}\sum_{(j\beta_{jk}>0)=i+1}^{n} q_{ij} x_{ik} x_{jk} \tag{7}$$

$$\sum_{(j|\beta_{jk}>0)=1}^{n} w_j x_{jk} + \sum_{(r|\sigma_{rk}>0)=1}^{h} s_r y_{rk} \le c_k, \quad \forall k \in K \tag{8}$$

$$\sum_{(k|\beta_{jk}>0)=1}^{m} x_{jk} \le 1, \quad \forall j \in J \tag{9}$$

$$\sum_{(j|\beta_{jk}>0)=1}^{n} t_{rj} x_{jk} \le U y_{rk} \quad \forall (r,k|\sigma_{rk}>0) \in R, \ k \in K \tag{10}$$

$$x_{jk} \in \{0,1\}, \quad \forall j \in J, \ k \in K \text{ with } \beta_{jk} > 0 \tag{11}$$

$$y_{rk} \in \{0,1\}, \quad \forall r \in R, \ k \in K \text{ with } \sigma_{rk} > 0 \tag{12}$$

where, $U$ is a large positive number.

The objective function (7) maximizes the total profit. Constraint (8) ensures that the capacity of any knapsack cannot be exceeded. Constraint (9) ensures every item can be selected for only one knapsack. Constraint (10) ensures that the decision variable yrk is equal to 1 if any item in every class is selected for knapsack $k$.

Constraint (6) guarantees that the knapsack usage of class r cannot exceed the available maximum number of knapsacks for this class. Finally, (11) and (12) are integrality constraints. After these additions, the model has $m+n+h+(h\times m)$ constraints and $(n\times m)+(h\times m)$ binary decision variables. Because of the NP-hardness of this problem, finding the optimal solution is not easy. Hence, heuristic methods should be used, especially for solving large-scale instances. The following section explains the proposed meta-heuristic methods for this model.

## 3. Proposed solution approaches

The developed G-QMKP model, especially if it is a large scale problem, cannot be solved easily, due to its NP-hard nature. Therefore, in this study, two different meta-heuristic solution approaches are suggested for solving the G-QMKP. The first is a genetic algorithm and the other is a hybrid algorithm which combines the F-MSG algorithm and the genetic algorithm. Previously, the MSG algorithm has been applied for solving the QKP by Saraç and Sipahioglu [25]. However, the MSG algorithm has two important disadvantages. These are determining the proper parameters for the algorithm and solving the unconstrained sub problem. The F-MSG algorithm was presented by [16] because it describes the value of the upper limit for updating step size parameters. which is one of the MSG parameters. Nevertheless, solving the dual problem is still an important hurdle. Actually, the motivation of the proposed hybrid algorithm is solving the sub problem using a genetic algorithm. In the following sections, these two proposed methods are explained, in detail.

### 3.1. A genetic algorithm for the G-QMKP (GAG-QMKP)

There are few studies in the literature for solving the QMKP with a genetic algorithm. Julstrom (2006) first introduced the QMKP and developed a GA for it. Saraç and Sipahioglu [25] and Singh and Baghel [26] proposed different GAs for the QMKP. In this study, a new GA that has similar representation and operators with the former study presented by Saraç and Sipahioglu, [25] is offered for the G-QMKP. The developed GA is as follows:

*Representation*: An n bit integer string is used to represent candidate solutions to the G-QMKP where the integer number {1,2, …,*k*} means the number of knapsacks, inclusion of item, and zero the exclusion of one of the n items from all of the knapsacks. For example, a solution for the 7-item, 2-knapsack problem can be represented as the following bit string: [0,2,0,1,0,0,2]. This means that items 2 and 7 are selected to be filled in knapsack two and item 4 is selected to be filled in knapsack one. This representation may yield an infeasible solution.

*Handling Constraints*: In the proposed method a modifying genetic operator strategy is chosen to generate feasible solutions which satisfy constraints (8)–(12). A penalizing strategy is used for satisfying constraint (6). So, $G_{AG\text{-}QMKP}$ generates solutions satisfying all the constraints of the G-QMKP.

*Initial population*: To generate a chromosome, the knapsacks are ordered according to their capacities and starting from the knapsack with the smallest capacity, each knapsack is filled with randomly selected items of smaller weights than the remaining knapsack capacity.

*Fitness function*: As $a_r$, r. being the slack variable added to the model in order to turn the constraint into an equation, for $G_{AG\text{-}QMKP}$ the below given number (13) fitness function was used.

$$\underset{x\in S}{Fitness_{GA\ G-QMKP}}(x,u,c)$$

$$=-\left(\sum_{(j|\beta_{jk}>0)=1}^{n}\sum_{k=1}^{m}p_{jk}x_{jk}+\sum_{(i|\beta_{ik}>0)=1}^{n-1}\sum_{(j|\beta_{jk}>0)=i+1}^{n}\sum_{k=1}^{m}q_{ij}x_{ik}x_{jk}\right)+penalty\sum_{r=1}^{h}\left(\sum_{(k|}\sigma_{rk}>0)=1^{m}y_{rk}+a_r-\eta_r\right) \quad (13)$$

*Genetic Operators*: The proposed genetic algorithm has three genetic operators; reproduction, crossover and mutation. They are explained below.

*Reproduction*: The 2-tournament selection method is used as reproduction operators.

*Crossover*: A specialized uniform based crossover operator is developed to maintain the feasibility of chromosomes for the constraints (8)–(12). Randomly selected two parent chromosomes are used to generate two offspring. A randomly selected gene of the first chromosome is interchanged with the corresponding gene that is placed in the same order of the second chromosome, if the capacities of the knapsacks are available for both of the chromosomes. If any capacity of a knapsack is exceeded by adding a new gene from the other parent chromosome, the value of this gene is changed to zero. The remaining capacity of the offspring is completed by selecting the excluded items with high profit.

*Mutation*: The developed GA contains two independent mutation operators each with its own mutation rate. Martello and Toth [18] proposed a polynomial time approximate algorithm for multiple knapsack problems with linear objective function. Saraç and Sipahioglu [25] modified the improvement techniques of this approximate algorithm for the QMKP. Both mutation operators use these techniques as explained below. Mutation 1 improves on the solution through local exchanges. First, it considers all pairs of items assigned to different knapsacks and, if possible and if the total profit increases, interchanges them. Mutation 2 removes s included items randomly. Where s is the parameter of mutation 2. Then the capacity of the chromosome is completed as in the crossover.

Since the elitism selection improves the efficiency of a GA considerably, as it prevents losing the best results, it is used in the developed GA. In this study, two types of termination conditions are used together. One of them checks whether the algorithm has run a fixed number (*nf*) of generations. The other stops the algorithm if the solution remains the same during $n_i$ generations. This proposed method was used to solve test problems and the results are given in Section 4.

### 3.2. An F-MSG based hybrid algorithm for the G-QMKP

The developed hybrid approach uses two different algorithms: F-MSG and GA. The F-MSG algorithm has an important disadvantage; solving the unconstraint sub problem. The hybrid approach implements the F-MSG algorithm basically as well as using the GA in order to obtain a solution for the sub problem. The F-MSG is an algorithm only for models with equality constraints. However, Sipahioglu and Saraç demonstrated in 2009 that the MSG algorithm could also be used for solving the QKP model with inequality constraint. This idea was also used in the hybrid solution approach proposed in this study.

The steps of the hybrid approach and related definitions are given below.

$f_0(x)$: objective function
$f(x)$: constraint functions

$k$: number of updates of dual variables
$n$: number of iterations
$u_k$ and $c_k$: values of dual variables in kth updates
$H_n$: upper limit value in $n$th iteration
$\varepsilon_1$ and $\varepsilon_2$: acceptable errors
$M$: The large positive number that will be used as the stopping tolerance while investigating the availability of a suitable solution smaller than or equal to the upper limit value. (While $l$ is a monotone increasing function, the search is stopped when the $l(k)$ value exceeds the value $M$.)
$\Delta_n$: Parameter of the positive step size used in updating the upper limit.
$\alpha$ and $\delta$: step size coefficients
$q$: Number of cases where the constraint achievement problem is solved and the constraints of the original problem are achieved.
$t$: Number of cases where constraint achievement problem could not be solved.

The steps of the algorithm for a problem having a minimum objective function are as follows:

*Step 1*: Assign positive numbers for $\varepsilon_1$, $\varepsilon_2$, $\Delta_0$ and M, and any number for $H_0$.
*Step 2*: Select $(u_1^n, c_1^n) \in R^p \times R_+$ and assign $0 < l(1) < M$ and $k=1$, $u_k = u_1^n$, $c_k = c_1^n$.
*Step 3*: Solve the following sub problem for given$(u_k, c_k)$.
Find $x \in S$ with the condition that P(Hn): $f_0(x) + c_k \| f(x) \| - \langle u_k, f(x) \rangle \le H_n$. If a solution that satisfies the constraint cannot be found (for instance if $l(k) > M$) go to Step 6. If the solution is found, check whether the constraints of the original problem are achieved. If $f(x_k^n) = 0$ (or $\|f(x_k) \le \varepsilon_1\|$) go to Step 5.
*Step 4*: Update dual variables using the below presented formulations:

$$u_{k+1} = u_k - \alpha\, s_k f(x_k),$$

$$c_{k+1} = c_k + (1+\alpha) s_k \| f(x_k) \|,$$

Here, $s_k$ is the positive step size parameter and it can be calculated with one of the formulations given below:

$$0 < s_k^1 = \frac{\delta\alpha\,(H_n - L(x_k, u_k, c_k))}{(\alpha^2 + (1+\alpha)^2)\,\| f(x_k) \|^2},$$

$$0 < s_k^2 = \frac{\delta(\alpha\,(H_n - L(x_k, u_k, c_k)) + (\bar{c} - c_k)\,\| f(x_k) \|)}{(\alpha^2 + (1+\alpha)^2)\,\| f(x_k) \|^2}$$

where, $\alpha > 0$, $0 < \delta < 2$ and $\bar{c} \ge c_k$. In addition, the step size parameters $s_k^1$ or $s_k^2$ have to meet the following condition for dual variables$(u_k, c_k)$.

$$s_k \| f(x_k) \| + c_k - \| u_k \| > l(k),$$

If this condition is not met, increase $c_k$ for having $c_k = \gamma\,(c_k + 1)$, $\gamma > 1$. At this point the $l(k)$ function is any function defined as when $k \to +\infty$, $l(k)$ also $\to +\infty$. Update $l(k)$ function, assign $k = k+1$ and go to *Step 3*.
*Step 5*: When $f(x_k) = 0$, $x_k$ is the solution of the $P(H_n)$ problem, and $L(x_k, u_k, c_k) = f_0(x_k)$. Assign $q = q+1$ and check t. If $t = 0$, assign $\Delta_{n+1} = \Delta_n$, and $t \ne 0$, then assign $\Delta_{n+1} = (1/2)\Delta_n$. Check the value of $\Delta_{n+1}$. If $\Delta_{n+1} \le \varepsilon_2$ then STOP. $f(x_k)$is an approximate optimal value, $x_k$ is an approximate solution of the original model, and $(u_k, c_k)$is an approximate solution of the dual problem. If $\Delta_{n+1} > \varepsilon_2$, assign $H_{n+1} = enk\{f_0(x_k),\ H_n - \Delta_{n+1}\}$, $n = n+1$ and go to *Step 2*.
*Step 6*: Assign t=t+1. If $q = 0$, assign $\Delta_{n+1} = \Delta_n$, if not assign$\Delta_{n+1} = (1/2)\Delta_n$. Assign $H_{n+1} = H_n + \Delta_{n+1}$, $n = n+1$ and go to *Step 2*.

The performance of the F-MSG algorithm on solving the G-QMKP actually depends on the performance of the method used for solving the sub problem in step 3. However, due to the fact that the model has a lot of constraints, solving the sub problem $(P(H_n))$ is not easy. Thus, obtaining a good solution in a reasonable time is really difficult, especially for large scale models. Therefore, a GA $(GA_{subproblem})$ is proposed and used to solve the sub problem.

The proposed genetic algorithm to solve the sub problem of the F-MSG algorithm $(GA_{subproblem})$ has similar representation, operators and terminating conditions with the GA approach described in Section 3.1. However, it has a different fitness function from the GAG-QMKP. The mathematical model for the G-QMKP contains six constraints. GAsubproblem generates solutions satisfying five of them (8)–(12). The remaining constraint (6) is included in the Lagrange function to construct the sub problem of the F-MSG algorithm. In this way, all the constraints of the model are considered and a feasible solution can be obtained. In other words, the $GA_{subproblem}$ generates the solutions in the set $S$ which is given in (14). The fitness function used for the $GA_{subproblem}$ is given in formulation number (15).

$$S = \left\{ \begin{array}{l} x \in S \Big| \sum\limits_{(j|\beta_{jk} > 0) = 1}^{n} w_j x_{jk} + \sum\limits_{(r|\sigma_{rk} > 0) = 1}^{h} s_r y_{rk} \le c_k \,\forall k \in K; \sum\limits_{(k|\beta_{jk} > 0) = 1}^{m} x_{jk} \le 1 \,\forall j \in J; \\ \sum\limits_{(j|\beta_{jk} > 0) = 1}^{n} t_{rj} x_{jk} \le U y_{rk} \,\forall (r, k|\sigma_{rk} > 0) \in R, k \in K; \\ x_{jk} \in \{0, 1\} \quad \forall (j, k|\beta_{jk} > 0) j \in J, k \in K \end{array} \right\} \tag{14}$$

$$\underset{x \in S}{Fitness_{GAsubproblem}}(x, u, c) = -\left( \sum_{(j|} \beta_{jk} > 0) = 1^n \sum_{k=1}^{m} p_{jk} x_{jk} + \sum_{(i|} \beta_{ik} > 0) = 1^{n-1} \sum_{(j|} \beta_{jk} > 0) = i + 1^n \sum_{k=1}^{m} q_{ij} x_{ik} x_{jk} \right) + c \sqrt{\sum_{r=1}^{h} \left( \sum_{(k|} \sigma_{rk} > 0) = 1^m y_{rk} + a_r - \eta_r \right)^2} - \sum_{r=1}^{h} u_r \left( \sum_{(k|} \sigma_{rk} > 0) = 1^m y_{rk} + a_r - \eta_r \right) \tag{15}$$

Proposed hybrid algorithm has certain important characteristics listed below:

- In the solution of the sub problem, a good starting upper limit value is not needed. Since the objective function is turned into minimization for solving the model with F-MSG, the upper limit can be indicated by 0 for all instances. The solution, since all the dual values will be zero when the algorithm is initially implemented, will be the solution of the original problem obtained through genetic algorithm. Moreover, it is expected to be a close value of the best value of the problem. In this way, the upper limit taken as 0 at first is updated to a successful upper limit value with the first iteration. To start with such an

upper limit and to focus on the success of the objective function, instead of focusing on satisfying the constraints, constitutes a significant decrease in the number of iterations.

- If the solutions of the original and the dual problem obtained in each iteration are not transferred to the following iterations, the genetic algorithm will always try to solve the same problem. In order to prevent this, the achieved best solution in any iteration is kept and is transferred to the initial population of the next iteration. In this way, the GA is able to resume the search. On the other hand, with similar logic, when the dual variables require updating, the GA continues from the values achieved in the previous run. By this means a fast improvement is enabled.
- When solving the problem using a genetic algorithm, the randomly generated initial population affects the result to be obtained. Working with big populations, using structures that enable creations of new solutions (such as mutation, immigration, and so on), and running the algorithm for multiple generations reduce this dependency. In general, when a GA is used, each problem is solved at certain times. However, in the proposed hybrid algorithm, since a new initial population that includes the best obtained solution is generated at each iteration, the need to solve the problem at certain times is significantly reduced.

## 4. Computational results

In order to test the performance of the proposed heuristic methods, randomly generated test instances were used. Since the G-QMKP has only recently been defined as a problem, the literature does not have any instances of it. For this reason, test problems were generated randomly within the frame of the established rules. The characteristics of the test problems (the way they were generated) and the obtained results, are given in the following sections. The proposed methods (GA, $GA_{subproblem}$ and HA) were coded using MS Excel/Visual Basic Application. All of the tests were implemented on a PC with a 2.8 GHz Intel Core i7 with 8 GB RAM. All of the test problems were also solved using the Gams/Dicopt solver to compare results.

### 4.1. Test instances

In the knapsack problems, the availability or unavailability of a correlation between the $w_j$ weight and pj profit parameters affect the difficulty of the problem. The problem becomes more difficult as the correlation increases [17]. Knapsack instances are usually generated without correlation, with weak correlation or with a strong correlation [19,2]. Hua and Huang [14] proposed a formulation for generating a quadratic knapsack problem without correlation, with weak correlation and with strong correlation. An important aspect of quadratic knapsack problems is the density, which is the percentage of the pj and qij profit parameters are other than zero. In the test problems of Billionnet and Soutif [4], which were also used by Julstrom [15], problems with densities of 25%, 75% and 100% were generated. Another important aspect is the magnitude of knapsack capacity. In the study of Hiley and Julstrom [13] capacity value was derived by utilizing the formula $c_1=c_2=\ldots.=c_m=(0.8(\sum_{j=1}^{n} w_j))/m$. On the other hand, in the QMKP problems derived by Billionnet and Soutif [4], it can be observed that the $c/\sum_{j=1}^{n} w_j$ rate varies between 3% and 93%.

In multiple knapsack problems, the number of knapsacks (m) is an important parameter. In their study, Hiley and Julstrom [13] solved the QMKP test problems with 100 and 200 jobs by considering cases where 3, 5 and 10 knapsacks were available.

The generalized quadratic multiple knapsack problem involves different decision variables and constraints from previous studies, for example, the number of item class, the number of items in any class, the maximum number of knapsacks which the items in class r can be selected for, the rate of items that can be selected for all knapsacks, and the level of preference of selected items in class $r$ to knapsack $k$.

The test problems used in this study were derived in a way that included all of these factors. The way to ensure this was as follows: The $w_j$, $p_j$ and $q_{ij}$ parameters in the test problems were designed in three different structures as uncorrelated, weakly correlated and strongly correlated, following Hua and Huang [14]. Table 1 shows how the parameters were generated as per their correlation levels. At this point, the notation $x \sim U(y,z)$ means that the random variable x is uniformly distributed within the $[y,z]$ range. The $\overline{w}$, $\overline{p}$ *and* $\overline{H}$ values in Table 1 are the constants taken from Hua and Huang [14] and are used for generating samples.

For the $\overline{w}, \overline{p}$ *and* $\overline{H}$ values, Hua and Huang [14] utilized 5 different levels, which are shown in Table 2.

The first line (no) in Table 2 shows the group number of the test problems to be generated, while the following lines show the $\overline{w}, \overline{p}$ *and* $\overline{H}$ values to be used in the groups. Derived parameters were used by rounding up to integers.

The trj parameter in G-QMKP model was determined as follows:

$$t_{r(j|j \le h)} = \begin{cases} 1 & if \quad r = j \\ 0 & otherwise \end{cases}$$

With this expression, in order to include at least one item for each class, the first h items starting from the 1st item, each assigns to the different classes ranging between *1* and *h*. Values of $t_{rj}$ parameters of $(j > h)$ items were designed so that they could be generated in two ways. This value may be zero or an integer with a random value within the range of [1, $h$] for each $j$ and $\pi_j$.

$$t_{r(j|j > h)} = \begin{cases} 1 & if \quad r = \pi_j \\ 0 & otherwise \end{cases}$$

If items are densely included in the same class, the trj parameter is generated as a random value within the range of $\pi$, [1, $h$], as shown below:

$$t_{r(j|j > h)} = \begin{cases} 1 & if \quad r = \pi \\ 0 & otherwise \end{cases}$$

The $s_r$ parameter, which shows the setups of class, and the items in class $r$ can be selected for a maximum of $\eta_r$ different knapsacks. On the other hand, the $\psi_{rk}$ parameter that shows knapsack preferences of the items in class $r$ was designed to enable four types of generation; all of the class having the same knapsack preferences,

**Table 1**
Sample generating parameter values taken from Hua and Huang [14].

| Uncorrelated | Weakly correlated | Strongly correlated |
|---|---|---|
| $w'_j \sim U(1,\overline{w})$ | $w'_j \sim U(1,\overline{w})$ | $w'_j \sim U(1,\overline{w})$ |
| $p'_j \sim U(0,\overline{p})$ | $p'_j = 10w_i \pm U(0,\overline{p}/10)$ | $p'_j = 10w_i + \overline{p}/10$ |
| $q'_{ij} \sim U(0, 4\overline{H})$ | $q'_{ij} = w_i/5 + w_j/5 \pm U(0, 2\overline{H})$ | $q'_{ij} = 2(w_i/5 + w_j/5)$ |

**Table 2**
Parameter levels used by Hua and Huang [14].

| No | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\overline{w}$ | 10 | 20 | 30 | 40 | 50 |
| $\overline{p}$ | 100 | 200 | 300 | 400 | 500 |
| $\overline{H}$ | 2 | 4 | 6 | 8 | 10 |

two groups of class having the same preferences, three groups of class having the same preferences and a class having completely different knapsack preferences.

The $\sigma$-ratio parameter is the rate of items that can be selected for all knapsacks. If any items can be selected for only one knapsack ($\sigma$-ratio=0), then the problem is reduced to k independent single knapsack problems. If any items can be selected for all knapsacks ($\sigma$-ratio=1), then the problem is a multiple knapsack problem.

As a result, it can be determined that there are 12 factors that affect the problem structure for generating a G-QMKP instance. These factors, their descriptions and the low and high levels of the parameters are given in detail in Table 3.

Even in the case where the 12 factors given in Table 3 are examined with only two levels, if a full factorial experiment is to be designed, 212=4096 different types of problems will have to be generated. For this reason, a 1/128 fractional design, where all the factors are double leveled, was selected and, by this means, it was determined that the generation of 32 type test problems would be sufficient. The properties of each of the problems are given in Table 4 in terms of the low (−) and high (+) levels of the factors and, in this study, three instances were generated for each type of problem. All the G-QMKP instances are available on the web site (http://endustri.ogu.edu.tr/Personel/Akademik_personel/Tugba_Sarac_Test_Instances/G-QMKP-instances.rar).

### 4.2. Test results

Each of the generated test problems was solved by means of the two proposed solution methods (GA and HA). In order to make a comparison, with the Gams (24.0.2)/Dicopt solver, which is used particularly in solving integer and nonlinear models, the running time is limited to 13,000 s. The obtained results of the instances with $n$=30 and $n$=300 are presented in Tables 5 and 7 respectively. In order to summarize the test results, two extra tables were added as Table 6 and 8.

The first 4 columns of Tables 5 and 7 show the properties of the test instances, such as the number of instances (*num*), the number of knapsacks ($k$), the number of class ($r$) and density (Percentage of non-zero $q_{ij}$) (d). Note that the number of instances has two parts in Tables 5 and 7. The first part shows the problem type in Table 4 and the second part shows the copy number of the problem. The fifth and the sixth columns in Tables 5 and 7 show the obtained objective value and the solution time using the Gams/Dicopt solver. The following three columns describe the obtained results with the genetic algorithm. (The obtained minimum objective value within three runs, the mean of the objective value of three runs and the solution time). The last two columns show the obtained objective value and the solution time using a hybrid algorithm. The best results are marked as bold and underlined for each instance.

It can be seen from Table 5 that the Gams/Dicopt solver obtained the best solutions in 35 of the 48 instances. On the other hand, the GA has 24 and the HA has 27 best solutions of the 48 test instances. Table summarizes the results shown in Table 5.

The first column of Table 6 shows the levels of $k$, $r$ and $d$. The following three columns describe the number of obtained best results of the Gams, GA and HA for each factor level. In this table, the number of best solutions is demonstrated for each solution method according to different parameter levels. Therefore, the effects of change in parameter level can be easily observed.

**Table 4**
Parameter Levels Table According to the 1/128 Fractional Factorial Experiments.

| sample | $n$ | *No* | Correlation | $k$ | $c$-ratio | $t_{rj}$ | $s_r$ | $\eta_r$ | $h$ | $\sigma$-ratio | $\psi_{rk}$ | Density |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | + | − | − | − | + | − | + | + | − | + | − | − |
| 2 | + | − | − | + | − | + | − | − | + | + | − | − |
| 3 | + | + | + | − | + | + | − | − | + | − | − | − |
| 4 | + | − | + | − | − | − | − | + | + | + | − | + |
| 5 | − | + | + | + | − | + | − | − | + | − | + | + |
| 6 | − | − | + | − | − | + | + | − | − | + | + | − |
| 7 | + | + | + | − | − | − | − | − | − | + | + | + |
| 8 | − | − | − | + | + | + | + | + | + | − | − | − |
| 9 | + | + | − | + | − | + | − | + | − | + | + | − |
| 10 | + | − | − | + | + | − | − | − | − | − | + | + |
| 11 | + | − | + | − | + | + | − | + | − | − | + | − |
| 12 | + | + | + | + | + | + | + | + | + | + | + | + |
| 13 | + | − | + | + | − | − | + | − | + | − | + | − |
| 14 | + | + | − | − | − | + | + | − | − | − | − | + |
| 15 | − | − | − | − | − | − | − | − | − | − | − | − |
| 16 | + | + | + | + | − | − | + | + | − | − | − | − |
| 17 | + | − | + | + | + | + | + | − | − | + | − | + |
| 18 | − | + | − | − | + | + | − | + | − | + | − | + |
| 19 | + | − | − | − | − | + | + | + | + | − | + | + |
| 20 | − | − | − | − | + | + | − | − | + | + | + | + |
| 21 | + | + | − | − | + | − | + | − | + | + | + | − |
| 22 | − | + | − | + | + | + | + | − | − | − | + | − |
| 23 | − | − | + | + | − | + | − | + | − | − | − | + |
| 24 | + | + | − | + | + | − | − | + | + | − | − | + |
| 25 | − | + | − | + | − | − | + | − | + | + | − | + |
| 26 | − | − | + | − | + | − | + | − | + | − | − | + |
| 27 | − | + | − | − | − | − | − | + | + | − | + | − |
| 28 | − | + | + | − | − | + | + | + | + | + | − | − |
| 29 | − | + | + | + | + | − | − | − | − | + | − | − |
| 30 | − | − | − | + | − | − | + | + | − | + | + | + |
| 31 | − | − | + | + | + | − | − | + | + | + | + | − |
| 32 | − | + | + | − | + | − | + | + | − | − | + | + |

**Table 3**
12 Factors affecting the structure of test problems and levels of these factors.

| Factor name | Description | Low level (−) | High level (+) |
|---|---|---|---|
| $n$ | Number of items | 30 | 300 |
| No | parameter levels in Table 2 | 1 | 5 |
| Correlation | correlation level between the parameters of wj and pj and pij | Uncorrelated | Strongly correlated |
| $k$ | number of knapsacks | $n$/30 | $n$/10 |
| $c$-ratio | the rate with which the capacity meets demand | 0.4 | 0.8 |
| $t_{rj}$ | items' distribution profile to classes | Random | One class dense |
| $s_r$ | setup time | [1,$\overline{w}$/4] | [$\overline{w}$/4, $\overline{w}$/2] |
| $\eta_r$ | maximum number of knapsacks which the items in class r can be selected for | 1 | [2, $n$/$h$] |
| h | number of item class | $n$/10 | $n$/2 |
| $\sigma$-ratio | rate of items that can be selected for all knapsacks | 0.4 | 0.8 |
| $\psi_{rk}$ | level of preference of selected item in class r to knapsack k, | Knapsack preferences of all classes are the same | Knapsack preferences of all classes are different |
| density | rate of $p_j$ and $p_{ij}$ parameters other than zero | 0.25 | 1.00 |

**Table 5**
Computational Results for Generated Test Instances with $n=30$.

| GAMS/DICOPT | | | | | | Genetic Algorithm | | | Hybrid Algorithm | |
|---|---|---|---|---|---|---|---|---|---|---|
| *num* | *k* | *r* | *d* | $z_G$ | $t\,(s)$ | $z_{GA\_max}$ | $z_{GA\_mean}$ | $t_{\_mean}(s)$ | $z_{HA}$ | $t\,(s)$ |
| 5-1 | 3 | 15 | 1.00 | **2835.30** | <1 | **2835.30** | 2768.18 | <1 | **2835.30** | 4 |
| 5-2 | 3 | 15 | 1.00 | **3304.80** | <1 | 3219.90 | 3211.77 | 2 | **3304.80** | 3 |
| 5-3 | 3 | 15 | 1.00 | 1634.20 | <1 | **1678.00** | 1630.17 | <1 | **1678.00** | 2 |
| 6-1 | 1 | 3 | 0.25 | **346.40** | <1 | **346.40** | 346.40 | <1 | **346.40** | 2 |
| 6-2 | 1 | 3 | 0.25 | **554.00** | <1 | **554.00** | 554.00 | <1 | **554.00** | 2 |
| 6-3 | 1 | 3 | 0.25 | **428.70** | <1 | **428.70** | 414.03 | <1 | **428.70** | 2 |
| 8-1 | 3 | 15 | 0.25 | 302.51 | <1 | **309.21** | 308.85 | 3 | **309.21** | 1 |
| 8-2 | 3 | 15 | 0.25 | 351.07 | <1 | **353.85** | 353.48 | <1 | **353.85** | 4 |
| 8-3 | 3 | 15 | 0.25 | 539.57 | <1 | **541.57** | 540.90 | <1 | **541.57** | 2 |
| 15-1 | 1 | 3 | 0.25 | **91.54** | <1 | **91.54** | 90.94 | <1 | **91.54** | 1 |
| 15-2 | 1 | 3 | 0.25 | 302.38 | <1 | **306.38** | 303.05 | <1 | **306.38** | 2 |
| 15-3 | 1 | 3 | 0.25 | 74.29 | <1 | **75.62** | 75.62 | <1 | **75.62** | 2 |
| 18-1 | 1 | 3 | 1.00 | **5387.70** | <1 | **5387.70** | 5387.70 | <1 | **5387.70** | 1 |
| 18-2 | 1 | 3 | 1.00 | **8551.08** | <1 | 8505.24 | 8407.48 | <1 | **8551.08** | 2 |
| 18-3 | 1 | 3 | 1.00 | **7760.51** | <1 | **7760.51** | 7760.51 | <1 | **7760.51** | 1 |
| 20-1 | 1 | 15 | 1.00 | **1599.85** | <1 | **1599.85** | 1592.77 | <1 | **1599.85** | 2 |
| 20-2 | 1 | 15 | 1.00 | 925.57 | <1 | **925.59** | 925.59 | <1 | **925.59** | 1 |
| 20-3 | 1 | 15 | 1.00 | **931.33** | <1 | 898.25 | 898.25 | <1 | **931,33** | 1 |
| 22-1 | 3 | 3 | 0.25 | **1923.61** | <1 | **1923.61** | 1589.02 | 2 | **1923.61** | 3 |
| 22-2 | 3 | 3 | 0.25 | **1314.09** | <1 | 884.40 | 884.40 | <1 | 884.40 | 1 |
| 22-3 | 3 | 3 | 0.25 | **1734.09** | <1 | 1680.09 | 1656.09 | <1 | 1680.09 | 2 |
| 23-1 | 3 | 3 | 1.00 | **471.00** | <1 | **471.00** | 462.80 | <1 | **471.00** | 2 |
| 23-2 | 3 | 3 | 1.00 | **955.70** | <1 | 379.60 | 379.03 | <1 | 381.60 | 1 |
| 23-3 | 3 | 3 | 1.00 | **1241.00** | <1 | 522.60 | 504.40 | 4 | 550.50 | 2 |
| 25-1 | 3 | 15 | 1.00 | 2111.63 | <1 | **2118.33** | 2021.70 | <1 | **2118.33** | 1 |
| 25-2 | 3 | 15 | 1.00 | **4262.64** | <1 | 3076.18 | 3076.18 | <1 | 3253,16 | 1 |
| 25-3 | 3 | 15 | 1.00 | **2951.90** | <1 | 2189.32 | 2054.60 | <1 | 2433,92 | 2 |
| 26-1 | 1 | 15 | 1.00 | **1747.60** | <1 | **1747.60** | 1730.8 | <1 | **1747.60** | 1 |
| 26-2 | 1 | 15 | 1.00 | **2433.60** | <1 | 1650.80 | 1596.53 | 2 | 1686 | 1 |
| 26-3 | 1 | 15 | 1.00 | **2293.20** | <1 | 1379.10 | 1375.07 | 2 | 1457,1 | 2 |
| 27-1 | 1 | 15 | 0.25 | 2245.95 | <1 | **2247.95** | 2247.95 | <1 | **2247.95** | 1 |
| 27-2 | 1 | 15 | 0.25 | **1966.52** | <1 | 1711.52 | 1711.52 | <1 | 1711,52 | 2 |
| 27-3 | 1 | 15 | 0.25 | **1383.49** | <1 | 1152.49 | 1152.49 | <1 | 1152,49 | 2 |
| 28-1 | 1 | 15 | 0.25 | 898.60 | <1 | **978.80** | 975.96 | <1 | **978.80** | 1 |
| 28-2 | 1 | 15 | 0.25 | **4024.70** | <1 | 2891.80 | 2843.17 | <1 | 3038.80 | 1 |
| 28-3 | 1 | 15 | 0.25 | **2634.00** | <1 | 1963.60 | 1867.87 | <1 | 1982.20 | 2 |
| 29-1 | 3 | 3 | 0.25 | 1903.80 | <1 | **1935.80** | 1563.88 | 2 | **1935.80** | 2 |
| 29-2 | 3 | 3 | 0.25 | 2820.00 | <1 | 2122.60 | 2122.20 | <1 | 2160 | 9 |
| 29-3 | 3 | 3 | 0.25 | **3283.60** | <1 | 2535.00 | 2436.20 | <1 | 2029,8 | 12 |
| 30-1 | 3 | 3 | 1.00 | **721.39** | <1 | **721.39** | 721.39 | <1 | **721.39** | 2 |
| 30-2 | 3 | 3 | 1.00 | **612.59** | <1 | 418.71 | 416.87 | <1 | 453,31 | 6 |
| 30-3 | 3 | 3 | 1.00 | **1032.35** | <1 | 763.19 | 763.19 | <1 | 774.99 | 7 |
| 31-1 | 3 | 15 | 0.25 | 477.90 | <1 | **491.90** | 487.80 | <1 | **491.90** | 2 |
| 31-2 | 3 | 15 | 0.25 | **594.00** | <1 | 544.00 | 542.53 | <1 | 547,6 | 4 |
| 31-3 | 3 | 15 | 0.25 | **508.60** | <1 | 435.40 | 430.63 | 3 | 435.8 | 3 |
| 32-1 | 1 | 3 | 1.00 | **11425.20** | <1 | 11254.90 | 11254.90 | <1 | 11254.90 | 1 |
| 32-2 | 1 | 3 | 1.00 | **15914.20** | <1 | 10820.80 | 10807.33 | <1 | 10820.80 | 5 |
| 32-3 | 1 | 3 | 1.00 | **19273.50** | <1 | 13087.80 | 12990.50 | 2 | 13087.80 | 3 |

**Table 6**
Number of the best results of test instances with $n=30$ for different parameter levels.

| Parameter | Level | GAMS | GA | HA |
|---|---|---|---|---|
| k | 1 | **19** | **13** | **15** |
| | 3 | 16 | 11 | 12 |
| r | 3 | 20 | 12 | 13 |
| | 15 | 15 | 12 | **14** |
| d | 0.25 | 13 | **14** | **14** |
| | 1 | **22** | 9 | 13 |

For instance, the Gams/Dicopt obtained 19 best solutions for the problems with one knapsack and obtained 16 best solutions for the problems with three knapsacks. Similarly, it reached 20 best solutions for the problems with three classes and 15 best solutions for the problems with 15 classes. As a result, it can be said that the Gams/Dicopt solver is more successful in the problems having a low number of knapsacks, a low number of classes and a high density. On the other hand, it can be said that the proposed GA and HA are not affected by the levels of parameters for the instances with $n=30$. Their success is almost the same for different $k$, $r$ and $d$ levels.

Computational results for generated instances with $n=300$ is given in Table 7.

It can be seen from Table 7, that the Gams/Dicopt solver obtained the best solutions in 20 of the 48 instances. On the other hand, the Gams/Dicopt solver did not give any feasible solution for 14 test instances within 13,000 s. This is a crucial result. Another important result is that the GA obtained the best results in just 6 of the 48 test instances. This success rate is very low. On the other hand, the proposed HA generated feasible solutions for all the test instances and obtained the best solution in 26 of the 48 test instances. With this result, it can be seen that the proposed HA is more successful than the other methods. Nevertheless, when evaluating the algorithms in terms of solution time, it can be seen that the genetic algorithm is better than the hybrid algorithm in

**Table 7**
Computational Results for Generated Test Instances with n=300.

| GAMS/DICOPT | | | | | | Genetic Algorithm | | | Hybrid Algorithm | |
|---|---|---|---|---|---|---|---|---|---|---|
| *num* | *k* | *r* | *d* | $z_G$ | *t* (s.) | $z_{GA\_max}$ | $z_{GA\_mean}$ | *t* (s.) | $z_{HA}$ | *t* (s.) |
| 1-1 | 10 | 30 | 0.25 | 4509.21 | 436.83 | 4948.45 | 4900.98 | 5960 | **4978,47** | 3023 |
| 1-2 | 10 | 30 | 0.25 | 4492.63 | 56.54 | **4889.58** | 4709.98 | 3431 | **4889.58** | 2142 |
| 1-3 | 10 | 30 | 0.25 | 5500.44 | 49.05 | 5653.92 | 5322.02 | 7038 | **5675.36** | 4821 |
| 2-1 | 30 | 150 | 0.25 | 2489.75 | 6058.47 | 2515.16 | 2487.90 | 3780 | **2524.39** | 5780 |
| 2-2 | 30 | 150 | 0.25 | 2098.35 | 5840.23 | 2181.44 | 2167.50 | 3264 | **2193.87** | 9741 |
| 2-3 | 30 | 150 | 0.25 | 2395.16 | 48.25 | 2494.78 | 2426.05 | 3274 | **2507.89** | 5117 |
| 3-1 | 10 | 150 | 0.25 | 27944.90 | 67.19 | 31854.80 | 31414.50 | 2100 | **31873.80** | 7000 |
| 3-2 | 10 | 150 | 0.25 | 37440.40 | 10.28 | **39356.00** | 32688.63 | 1976 | **39356.00** | 7333 |
| 3-3 | 10 | 150 | 0.25 | 28050.10 | 38.69 | 31981.50 | 31804.37 | 3212 | **32236.40** | 7274 |
| 4-1 | 10 | 150 | 1.00 | - | 13000.11 | 8126.80 | 7893.26 | 1680 | **8172.40** | 2260 |
| 4-2 | 10 | 150 | 1.00 | - | 13000.05 | 7216.50 | 7135.17 | 1765 | **7457.80** | 2265 |
| 4-3 | 10 | 150 | 1.00 | **8333.80** | 117.37 | 7114.70 | 6952.93 | 1347 | 7214.20 | 2082 |
| 7-1 | 10 | 30 | 1.00 | **67020.40** | 7843.08 | 63792.80 | 62929.70 | 1360 | 64415.40 | 3040 |
| 7-2 | 10 | 30 | 1.00 | **64591.70** | 7669.09 | 61044.00 | 60574.73 | 1346 | 61177.40 | 2684 |
| 7-3 | 10 | 30 | 1.00 | - | 13000.97 | 64813.50 | 64313.43 | 1413 | **64905.60** | 3879 |
| 9-1 | 30 | 30 | 0.25 | 8954.74 | 130.70 | 8784.74 | 8561.80 | 2320 | **9098.78** | 5880 |
| 9-2 | 30 | 30 | 0.25 | 12299.40 | 1778.74 | 12393.32 | 12172.31 | 2945 | **12440.80** | 6298 |
| 9-3 | 30 | 30 | 0.25 | **15612.84** | 1515.52 | 15326.97 | 15255.75 | 2941 | 14806.02 | 5559 |
| 10-1 | 30 | 30 | 1.00 | **15256.01** | 13000.08 | 12849.05 | 8541.28 | 825 | 12002.24 | 1080 |
| 10-2 | 30 | 30 | 1.00 | **13058.94** | 9667.52 | 12776.99 | 8297.00 | 1928 | 11192,08 | 1897 |
| 10-3 | 30 | 30 | 1.00 | - | 13000.03 | **12353.05** | 12220.50 | 3346 | 11920,80 | 1932 |
| 11-1 | 10 | 30 | 0.25 | **7057.00** | 1.81 | 6740.60 | 6636.74 | 2500 | 6694.30 | 2480 |
| 11-2 | 10 | 30 | 0.25 | **6617.00** | 1.75 | 6515.70 | 6465.30 | 1433 | 6460.9 | 2942 |
| 11-3 | 10 | 30 | 0.25 | **7677.30** | 0.89 | 7256.10 | 7250.70 | 2445 | 7227,30 | 4474 |
| 12-1 | 30 | 150 | 1.00 | - | 13006.01 | 56769.50 | 56250.00 | 5280 | **57789.00** | 2540 |
| 12-2 | 30 | 150 | 1.00 | - | 13000.01 | 56371.50 | 56009.37 | 6051 | **58729,40** | 2870 |
| 12-3 | 30 | 150 | 1.00 | - | 13000.02 | 56489.60 | 56331.80 | 4851 | **58352,20** | 3940 |
| 13-1 | 30 | 150 | 0.25 | 4093.20 | 90.56 | 4039.10 | 3941.60 | 620 | **4102.40** | 960 |
| 13-2 | 30 | 150 | 0.25 | **4047.50** | 242.54 | 4003.40 | 3950.20 | 742 | 4022,90 | 983 |
| 13-3 | 30 | 150 | 0.25 | **4582.90** | 1846.65 | 4563.90 | 4496.57 | 633 | 4339,60 | 957 |
| 14-1 | 10 | 30 | 1.00 | **26456.79** | 1.29 | 25417.99 | 24095.60 | 1500 | 25663.14 | 2280 |
| 14-2 | 10 | 30 | 1.00 | **25345.55** | 17.13 | 24345.72 | 23480.47 | 1136 | 23259,91 | 4643 |
| 14-3 | 10 | 30 | 1.00 | **30947.20** | 7.04 | 30571.69 | 30329.40 | 1066 | 30539,18 | 2559 |
| 16-1 | 30 | 30 | 0.25 | - | 13000.87 | 13396.70 | 13233.20 | 820 | **13408.10** | 2580 |
| 16-2 | 30 | 30 | 0.25 | - | 13000.08 | **16059.70** | 15672.77 | 654 | **16059.70** | 1539 |
| 16-3 | 30 | 30 | 0.25 | - | 13000.49 | **13646.50** | 13260.37 | 768 | **13646.50** | 1155 |
| 17-1 | 30 | 30 | 1.00 | **4096.90** | 219.13 | 2805.10 | 1547.82 | 5740 | 2722.50 | 2060 |
| 17-2 | 30 | 30 | 1.00 | **3837.70** | 243.42 | 2395.80 | 2058.70 | 5295 | 2760,30 | 3306 |
| 17-3 | 30 | 30 | 1.00 | **3573.80** | 173.32 | 2251.80 | 2176.67 | 4716 | 2499,20 | 4642 |
| 19-1 | 10 | 150 | 1.00 | **6824.12** | 22.40 | 6644.54 | 6590.27 | 1480 | 6548.03 | 2700 |
| 19-2 | 10 | 150 | 1.00 | **7920.00** | 185.06 | 7842.99 | 7766.54 | 1071 | 7855,19 | 2128 |
| 19-3 | 10 | 150 | 1.00 | **8082.10** | 38.10 | 8068.88 | 7994.47 | 1194 | 8080.79 | 1602 |
| 21-1 | 10 | 150 | 0.25 | - | 13000.02 | 20639.31 | 20373.40 | 2820 | **21101.37** | 4240 |
| 21-2 | 10 | 150 | 0.25 | 23159.36 | 8025.94 | **23905.12** | 23530.46 | 2629 | **23905.12** | 4176 |
| 21-3 | 10 | 150 | 0.25 | 22238.50 | 1422.14 | 22673.42 | 22428.85 | 2765 | 21601,18 | 4984 |
| 24-1 | 30 | 150 | 1.00 | - | 13000.00 | 48901.01 | 48373.30 | 3038 | **49766.25** | 4960 |
| 24-2 | 30 | 150 | 1.00 | - | 13000.00 | 52927.40 | 52526.90 | 2920 | **53386,40** | 5321 |
| 24-3 | 30 | 150 | 1.00 | - | 13000.08 | 45992.96 | 45906.30 | 3302 | **47810,15** | 5302 |

**Table 8**
Number of the Best Results of Test Instances with n=300 for Different Parameter Levels.

| Parameter | Level | GAMS | GA | HA |
|---|---|---|---|---|
| k | 10 | **12** | 3 | 11 |
| | 30 | 8 | 3 | **15** |
| r | 30 | **14** | **4** | 9 |
| | 150 | 6 | 2 | **17** |
| d | 0,25 | 6 | **5** | **17** |
| | 1 | **14** | 1 | 9 |

most of the test instances. This means there is a tradeoff between the solution time and the objective function value. If the solution time is not so important, a decision maker should choose the hybrid algorithm to solve this problem. Table 8 summarizes the results in Table 7.

The first column of Table 8 shows the levels of *k*, *r* and *d*. The following three columns describe the number of obtained best results of the Gams, GA and HA for each of the factor levels. In this table, the number of best solutions is demonstrated for each solution method according to different parameter levels. For instance, the Gams/Dicopt obtained 12 best solutions of the 20 test problems for one knapsack and obtained 8 best solutions for three knapsacks. Similarly, it reached 14 best solutions for three classes and 6 best solutions for 15 classes. As a result, it can be said that the Gams/Dicopt solver is more successful in problems having a low number of knapsacks, a low number of classes and a high density. These results are similar to obtained results with $n=30$. On the other hand, it can be said that the proposed HA is more successful in problems having a high number of knapsacks, a high number of classes and a low density. For the proposed GA, it can be said that it is more successful in problems having a low number of classes and a low density.

In order to show the performance of the HA in finding the optimum solutions of the instances, 10 very small instances with 10 items and two knapsacks were generated randomly. To find the optimum solutions of the instances, an enumeration algorithm

**Tablo 9**
Test results of the instances with 10 items and 2 knapsacks.

| Enumeration | | GAMS/ Dicopt | | Genetic Algorithm | | | Hybrid Algorithm | |
|---|---|---|---|---|---|---|---|---|
| num | $z_{opt}$ | $z_G$ | $t$ (s) | $z_{GA\_max}$ | $z_{GA\_mean}$ | $t\_mean$ (s.) | zHA | $t$ (s.) |
| 10_1 | 180,80 | **180,80** | < 1 | 180,80 | **180,80** | < 1 | **180,80** | < 1 |
| 10_2 | 265,00 | **265,00** | < 1 | 265,00 | **259,32** | < 1 | **265,00** | < 1 |
| 10_3 | 82,40 | **82,40** | < 1 | 82,40 | **82,40** | < 1 | **82,40** | < 1 |
| 10_4 | 148,80 | **148,80** | < 1 | 148,80 | **148,80** | < 1 | **148,80** | < 1 |
| 10_5 | 221,20 | **221,20** | < 1 | 221,20 | **221,20** | < 1 | **221,20** | < 1 |
| 10_6 | 296,20 | **296,20** | < 1 | 296,20 | **296,20** | < 1 | **296,20** | < 1 |
| 10_7 | 282,90 | **282,90** | < 1 | 278,80 | **270,40** | < 1 | **282,90** | < 1 |
| 10_8 | 101,80 | **101,80** | < 1 | 101,80 | **96,07** | < 1 | **101,80** | < 1 |
| 10_9 | 320,00 | **320,00** | < 1 | 320,00 | **320,00** | < 1 | **320,00** | < 1 |
| 10_10 | 237,40 | **237,40** | < 1 | 237,40 | **237,40** | < 1 | **237,40** | < 1 |

was developed and coded using MS Excel VBA. All of these instances were solved using enumeration, GAMS/Dicopt, GA and HA. The obtained results are given in Table 9.

The first column of Table 9 shows the number of instances (*num*), while the second column shows the optimum objective value found by enumeration. The third and the fourth columns show the obtained objective values and solution times using the Gams/Dicopt solver. Similarly, the following three columns describe the obtained results with the genetic algorithm and the last two columns show the obtained objective values and solution times using the hybrid algorithm. The optimum results are marked as bold and underlined for each instance.

It can be seen from Table 9 that both the GAMS/Dicopt and the HA obtained optimum solutions for all instances. However, for only one instance, the GA could not find the optimum solution (number 10_7). These results mean that HA can successfully find the optimum solutions for small instances.

## 5. Case study: plastic production with injection

As shown in Section 4, using the proposed hybrid approach, good results were obtained for the G-QMKP test instances. To show the performance of the proposed model on a large scale problem, a real life problem for a plastic production company was considered.

One of the methods of producing plastic parts is injection. Plastic parts produced through injection have a large market worldwide. For this reason, it is a sector with significant importance. With the injection method, plastic that is heated to a particular temperature is melted and injected into a mold with a certain level of pressure. After a fixed period of time, the piece is kept for cooling, the mold is opened and the piece is removed from it. Injection machine consists of two sections, the injection system and the clamping system. The injection system has a raw material chamber. This system serves to heat and melt the raw material and inject it into the mold at certain levels of pressure and rate. The clamping system on the other hand, is the moving section to which mold is fixed and with which to mold is closed and opened. The distance between the cylinders on which the clamping system moves is called the column spacing. The main parts of an injection machine are shown in Fig. 1.

The production of a plastic part with an injection machine is a single-stage production. It is possible to produce products of different raw materials and colors with an injection machine. In order to produce a plastic product in an injection machine, the relative mold has to be fixed to the machine and the mixture of raw material and paint of the desired color have to be introduced into the raw material chamber of the machine. Products having the same form but using different raw materials or colors are considered as different products. Therefore, in such facilities, it is possible to have different products using the same mold. Switching from the production of a certain product to another one requires setup times. Setup times cover the change of molds or the preparation and introduction of raw materials and paint mixtures, if necessary. Thus, while setup times are short for similar jobs, they can be longer for jobs that are dissimilar. If similar jobs are not assigned to the same machines, the total setup time needed for production may increase significantly. In addition, the jobs assembled together should also be assigned to the same machines as often as possible, to reduce the material handling cost. On the other hand, plastic part manufacturers are mostly on a sub-contract basis and therefore have to meet customer demands with as minimum a delay as possible. For this reason, production planning is usually made for the short term and delays in delivery time eliminated by assigning similar jobs and the job assemblies together to the same machines.

If injection machines are considered as knapsacks and jobs are considered as items, the problem of determining the jobs to be performed within a given planning period and of deciding with which machines the production should be conducted can be considered as a multiple knapsack problem. Yet, since it is desired to assign similar jobs to the same machine, there will be additional profits by assigning two jobs assembled together to the same machine, so the nature of the problem is quadratic. The capacities of knapsacks ($c_k$) are the working times of the injection machines within the related planning period. When the job i is selected for production, it provides a profit ($p_i$) at the rate of the closeness of its delivery time and if the jobs i and j are assigned together to the same machine and if they are assembled together, they provide an additional profit ($q_{ij}$). When the job i is assigned, it uses the production time of ($w_i$) from the capacity of the machine. The objective here is to determine the job subset that will maximize total profit without exceeding the ($c_k$) production times of the machines, and to determine which machines will be assigned to these jobs. Special constraints of the G-QMKP are also suitable for this problem as explained below:

- *Setup Constraint*: When a job is assigned to a machine, the related mold has to be attained to the same machine. Furthermore, not only the time necessary for production, but also the time needed for setup has to be taken into consideration in terms of capacity utilization. Jobs using the same molds belong to the same class and if any two or more jobs belonging to the same class are assigned to the same machine only one setup is needed, since all of these jobs use the same mold.
- *Assignment Conditions*: All molds may not be fixed to all machines due to technical conditions. For instance, for a mold to be fixed to the injection machine, its width and length have to be shorter than the column space and the mold depth has to be appropriate for the closing space. As a result, all molds may not be selected for every machine. Additionally, the maximum number of machines which the jobs in class r can be assigned to may be limited. Therefore, the mold can be assigned different machines at the same time if it has copies.
- *Knapsack preferences of items*: Even if certain machines seem to be carrying out exactly the same jobs, it may still be preferential for a given product to be assigned to certain machines for various reasons. (For instance, a mold may not work at the same efficiency with every machine it is fixed to, or the decision maker may have preferences on a job/mold to be assigned to a certain machine). In this case, the parameters related with the preferences can be described as follows:

  $\psi_{rk}$: Level of preference of assigning jobs in class r to machine k, [0,1]. (This parameter indicates how much

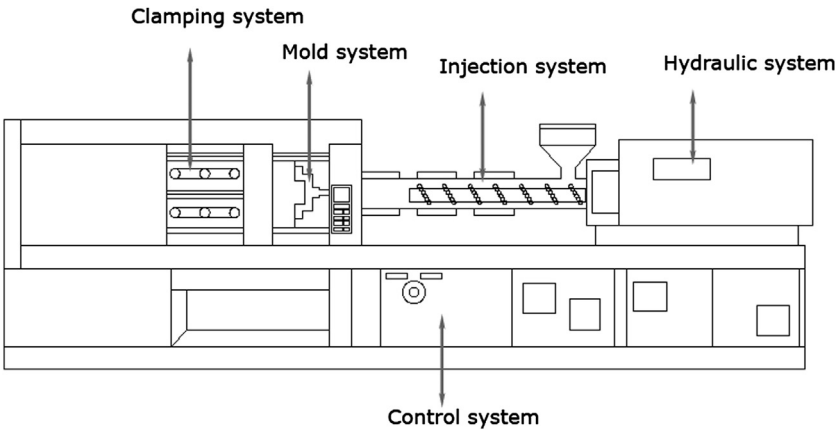


**Fig. 1.** Schematic display of an injection machine

**Table 10**
Test results for the real-life problem with $n=500$.

| GAMS/DICOPT | | GAG-QMKP | | | HA | | |
|---|---|---|---|---|---|---|---|
| z | $t$ (sn.) | $nf/ni$ | $z$ | $t$ (sn) | nf/ni | $z$ | $t$ (sn) |
| – | – | 1200/150 | 10926.87 | 58142 | 150/50 | **11420.87** | 48945 |
| – | – | 50/30 | 10632.65 | 3491 | 50/30 | 11128.00 | 36555 |
| – | – | 10/10 | 9718.39 | 783 | 10/10 | 10542.201;* | 4596 |

* A maximum of 5 iterations limit was set for HA.

the assignment of a job of a given class is preferred. It has a value of between 0 and 1. Value 1 indicates a high preference while 0 indicates that the assignment of a job of $r$. class to the $k$. machine is not preferred at all).
$p_j$: $j$. priority of the job
$p_{jk}$: preference of assigning job $j$ to machine k
The $p_{jk}$ parameter can be calculated with the following formulation:

$$p_{jk} = p_j \times \sum_{(r|\sigma_{rk}=1)=1}^{h} t_{rj}\psi_{rk}$$

In order to test the performance of the proposed algorithms on a large scale problem, a real life problem involving 500 jobs was used. A solution to this problem was attempted by the Dicopt solver of Gams, and solved by the proposed GA and HA. The obtained results are presented in Table 10.

The first two columns of Table 10 aim to show the objective function value (z) of the best solution that can be obtained within 13,000 s with the Gams/Dicot and solution time. However, the Dicopt could not produce a solution within the given time. In the third, fourth and the fifth columns of Table 10, the maximum number of generations allowed without improvement when running the GA ($nf/ni$), objective function value ($z$) and the time needed for solution are presented. In the last three columns, the maximum number of generations allowed without improvement when running the HA ($nf/ni$), objective function value ($z$) and the time needed for solution are displayed.

This real life problem is really large due to the number of jobs (500). Its mathematical model has 12,840 constraints and 44000 0-1 decision variables. The qij parameter itself only has 124,750 values in this model. It is not easy to solve such a large, nonlinear problem that involves 0–1 integer variables. In fact, the problem could not be solved with the Gams/Dicopt within 13,000 s. Indeed, it is deemed a success to even find a feasible solution to a problem of such difficulty. On the other hand, this problem was solved with the proposed GA and HA using three different generation number combinations. The most successful solution to the problem was achieved by means of the HA in approximately 13.5 h using large generation numbers. However, it was also demonstrated that if the solution quality can be slightly ignored, this problem can be solved in about 13 min with the GA and in less than one and a half hour with the HA. Furthermore, it is obvious that these times can be further reduced with good coding skills and by using fast programming languages such as C++.

## 6. Conclusions

In this study the generalized quadratic knapsack problem has been first introduced and a mathematical model was given. Due to the fact that the mathematical model developed for the G-QMKP is NP-hard, two different heuristic approaches were proposed. The first is a genetic algorithm, and the second is a hybrid algorithm. The hybrid algorithm was developed by the hybridization of the genetic algorithm with a method known in the literature as the F-MSG. In addition, an approach that can be used for generating test instances for the G-QMKP was proposed and the success of the proposed approaches were demonstrated on these randomly generated test instances. According to the results, it was determined that in terms of objective function value, the hybrid algorithm is superior, while in terms of solution time the genetic algorithm is better. Furthermore, it was also demonstrated that the problems for which no feasible solution can be obtained using the Gams/Dicopt solver, can be solved with the proposed algorithms, especially for large scale problems. Additionally, a case study in plastic production with injection was also conducted. Particularly with the GA, it was determined that a real-life problem involving 500 jobs can be solved in 15 min, considered to be a reasonable time.

In future research, to solve the sub problem of the F-MSG algorithm, more efficient algorithms could be developed or different meta-heuristics could be used. By these means, the time required to find a suitable solution would be reduced. Additionally, by re-coding the developed approaches with C++, solution times can be reduced. Moreover, the success of the hybrid algorithm may be tested on different NP-hard problems as a new solution approach.

## References

[1] Altay N, Robinson Jr PE, Bretthauer KM. Exact and heuristic solution approaches for the mixed integer setup knapsack problem. European Journal of Operational Research 2008;190:598–609.
[2] Anagun AS, Saraç T. Optimization of performance of genetic algorithm for 0-1 knapsack problems using Taguchi method. Lecture Notes in Computer Science 2006;3982:678–87.

[3] Billionnet A, Calmels F. Linear programming for the 0-1 quadratic knapsack problem. European Journal of Operational Research 1996;92:310–25.
[4] Billionnet A, Soutif E. An exact method based on Lagrangean decomposition for the 0-1 quadratic knapsack problem. European Journal of Operational Research 2003;157(3):565–75.
[5] Caprara A, Pisinger D, Toth P. Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing 1999;11:125–37.
[6] Caserta M, Quiñonez Rico E, Márquez Uribe A. A cross entropy algorithm for the Knapsack problem with setups. Computers and Operations Research 2008;35:241–52.
[7] Chaillou P, Hansen P, Mahieu Y. Best network flow bound for the quadratic knapsack problem. Combinatorial Optimization of Lecture Notes in Mathematics 1986;1403:225–35.
[8] Gallo G, Hammer PL, Simeone B. Quadratic knapsack problems. Mathematical Programming Study 1980;12:132–49.
[9] Gasimov R. Augmented Lagrangean duality and nondifferantiable optimization methods in non-convex programming. Journal of Global Optimization 2002;24:187–203.
[10] Gasimov RN, Ustun O. Solving the quadratic assignment problem using F-MSG algorithm. Journal of Industrial and Management Optimization 2007;3(2):173–91.
[11] Gen M, Cheng R. Genetic algorithms and engineering design. New York: John Wiley & Sons; 1997.
[12] Goldberg DE. Genetic Algorithms in Search, Optimization, and Machine Learning. USA: Addison–Wesley; 1989.
[13] Hiley A, Julstrom BA. The Quadratic multiple knapsack problem and three heuristic approaches to it. In: Proceedings of the genetic and evolutionary computation conference; 2006 p. 547–52.
[14] Hua ZS, Huang FH. A variable-grouping based genetic algorithm for large-scale integer programming. Information Science 2005;176(19):2869–85.
[15] Julstrom BA. Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In: Proceedings of the genetic and evolutionary computation conference, vol. no. 1; 2005, p. 607–14.
[16] Kasimbeyli R, Ustun O, Rubinov A. The modified subgradient algorithm based on feasible values. Optimization 2009;58(5):535–60.
[17] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. New York: Springer-Verlag; 2004.
[18] Martello S, Toth P. Knapsack Problems: algorithms and computer implementations. Chichester, UK: John Wiley&Sons; 1990.
[19] Martello S, Pisinger D, Toth P. New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operation Research 2000;123:325–32.
[20] McLay LA, Jacobson SH. Algorithms for the bounded set-up knapsack problem. Discrete Optimization 2007;4:206–12.
[21] Michel S, Perrot N, Vanderbeck F. Knapsack problems with setups. European Journal of Operational Research 2009;196:909–18.
[22] Michelon P, Veuilleux L. Lagrangean methods for the 0-1 quadratic knapsack problem. European Journal of Operational Research 1996;92:326–41.
[23] Pisinger D. The quadratic knapsack problem-a survey. Discrete Applied Mathematics 2007;155:623–48.
[24] Sang MS, Sang WL. A memetic algorithm for the quadratic multiple container packing problem. Applied Intelligence 2012;36:119–35.
[25] Saraç T, Sipahioglu A. A genetic algorithm for the quadratic multiple knapsack problem. Lecture Notes in Computer Science 2007;4729:490–8.
[26] Singh A, Baghel AS. A new grouping genetic algorithm for the quadratic multiple knapsack problem. Evolutionary Computation in Combinatorial Optimization, in the series Lecture Notes in Computer Science 2007;4446:210–8.
[27] Sipahioglu A, Saraç T. The performance of the modified subgradient algorithm on solving the 0-1 quadratic knapsack problem. Informatica 2009;20(2):1–12.
[28] Sundar S, Singh A. A swarm intelligence approach to the quadratic multiple knapsack problem. Neural Information Processing: Theory And Algorithms 2010;6443:626–33.
[29] Wang H, Kochenberger G, Glover F. A computational study on the quadratic knapsack problem with multiple constraints. Computers and Operations Research 2012;39:3–11.
[30] Xie XF Liu J. A mini-swarm for the quadratic knapsack problem. In: Proceedings of the IEEE Swarm Intelligence Symposium (SIS). Honolulu, HI, USA; 2007. p. 190–7.