Bare Demo of IEEEtran.cls for IEEE Conferences

Michael Shell
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: http://www.michaelshell.org/contact.html

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678–2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—The abstract goes here.**

I. Background

A. Hyper-heuristic

B. Problem Definition

The problem that we want to solve is the problem of generating personalized healthy menu recommendations for a person based on: (i) a set of available food packages published by different providers in a food marketplace, (ii) a set of person defined constraints (i.e. the preferred price for the generated menu, the delivery time for the generated menu, etc.), and (iii) a nutrition daily recommendation computed based on the person's profile (e.g age, weight, height, etc.). We approach this problem as an optimization problem and propose a Choice Function-based Constructive Hyper-Heuristic to solve it. Consecutively, we define the main concepts of our optimization problem as follow.

Search Space. In our approach, the *search space* is represented by a sets of food packages from different food providers available in food marketplace. For each of the food packages the providers need to specify the following information: the time of delivery and preparation for food package, the total cost of the food package, the meal type (e.g.breakfast, first snack, lunch, second snack, dinner), the set of nutrients values corresponding to the food package, the food provider name, and the food package rating (rating regarding the aspect, taste and smell of a meal given by clients that have already ordered this food package).

Solution of the optimization problem. A solution of the optimization problem is represented, in our approach, by a set of food packages corresponding to each meal of the day (i.e. breakfast,lunch, dinner, snacks).

$$sol = \{meal_{breakfast}, meal_{snack_1}, meal_{lunch}, meal_{snack_2}, meal_{dinner}\} \quad (1)$$

where a *meal* is composed of a set of food items corresponding to a meal of a day; additionally, for each of the meal of the day the following information is provided: (i) the total cost of the meal, (ii) the time of delivery and preparation for the meal, $(iii)$the meal rating (aspect, taste and smell of a meal) given by other persons, (iv) the meal type (e.g. breakfast,lunch, etc.), $(v)$the set of nutrients values corresponding to each food item that is part of the meal, and $(vi)$the food provider. Each solution is generated for a specific user, and the profile of each user is defined as:

$$profile = (personalInfo, personalConstr, dietRec) \quad (2)$$

where *personalInfo* includes information regarding the users name, sex, age, height, weight, and physical activity performed, *personalConstr* includes information regarding the price that the user is willing to pay for a menu per day, the delivery time in which the user wants its menu to be delivered, and weights given by the user which indicate the degree of importance for the aspect, taste and smell of the food package, and *dietRec* includes information regarding the daily kilo-calories needs, daily energy needs, and the set of nutrients values recommended for this user according to its personal profile.

Fitness Function. To evaluate the quality of a domain solution we have used a fitness function (see Formula **??**) we previously introduced in []. The fitness function will take all the aspects of a solution and compare them with the reference (the reference value is different for each user, it is created based on the user medical recommendations and preferences), the closer the solution is to the reference value the better it is. The lower the value of the fitness function the better the solution.

$$F_{final}(sol) = F(sol) + F_D(sol) \quad (3)$$

In formula 3, $F(sol)$ is defined as:

$$F(sol) = w_p * F_P(sol) + w_t * F_T(sol) + w_n * F_N(sol) + w_r * F_R(sol) \quad (4)$$

where:a) w_p, w_t, w_n, w_r are the weights which specify the importance of each component of the fitness function; the sum of these weights is one, (b)$F_P(sol)$ is a function which evaluates how close the total price of the menu (it

calculates the price of each meal) is to the value specified by the person; (c) $F_T(sol)$ is a function which evaluates how close the total preparation and delivery time of the menu(it calculates the preparation and delivery time of each meal) is to value specified by the person, (d)$F_N(sol)$ is a function which evaluates how close the values of the nutritional components are compared to the recommended values, (e) $F_R(sol)$ is a function which evaluates the ratings of the meals in the solution; elders have possibility to give a rating to the meals between 1 and 5 (1 is the lowest and 5 is the highest), there are three types of ratings: aspect, taste, smell.

$F_P(sol)$ is computed using the following formula:

$$F_P(sol) = \frac{\sum_{i=1}^{5} price(meal_i) - refPrice}{refPrice} \quad (5)$$

where the sum will go through the five meals, using the function $price$ which evaluates the price of a meal, and the $refPrice$ is the maximum price suggested by the user. The F_P function can return both negative, if the price of the solution is lower than the maximum suggested price(a good result), and positive values, if the price of the solution is higher than the maximum suggested price, it can also return the value 0 if the price is equal to the suggested price.

$F_T(sol)$ is computed using the formula bellow:

$$F_T(sol) = \frac{\sum_{i=1}^{5} time(meal_i) - referenceTime}{refTime} \quad (6)$$

where the sum will go through the five meals, using the function $price$ which evaluates the preparation and delivery time of a meal, $refTime$ is the maximum time suggested by the user. The F_T function can return both negative and positive values like the F_P function.

$F_N(sol)$ is as defined as:

$$F_N = \sum_{i=1}^{5} \sum_{j=1}^{n} \frac{abs(refValue_{meal_i,j} - nValue(meal_i, j))}{refValue_{meal_i,j}} \quad (7)$$

where n is the number of nutritional components and $refValue_{i,j}$ is the optimal value suggested for $meal_i$, $nutritionalComponent_j$. Function $nValue$ calculates the actual $nutritionalComponent_j$ value for $meal_i$. The function F_N is different from function F_T and function F_P, the results needs to be as close to the suggested value as possible, a lower value isn't a better one in this case.

$F_R(sol)$ is defined as:

$$F_R(sol) = \sum_{i=1}^{5} \alpha_{aspect} * (3 - aspectRating_{meal_i}) + \alpha_{taste} * (3 - tasteRating_{meal_i}) + \alpha_{smell} * (3 - smellRating_{meal_i}) \quad (8)$$

where α_{aspect}, α_{taste}, α_{smell} is the weight of each type of rating and $aspectRating$, $tasteRating$, $smellRating$ are the actual grades. The average rating is 3, if the user gives the average rating the F_R will have no effect on the F_{final} function.

$F_D(sol)$ from formula 3 evaluates how diverse the solution is, it compares the meals one with each other to see how many if there are ingredients that repeat themselves. This function takes values between 0 and 1, it works as a penalty for the F_{final} function, it doesn't have a weight assigned to it.

$$F_D = Min(Fruits_D(sol), Vegetables_D(sol), Others_D(sol), 1) \quad (9)$$

The functions $Fruits_D$, $Vegetables_D$ and $Others_D$ return the penalty based on the number of ingredient occurrences for Fruits, Vegetables and Cheese/Pasta.

II. Choice Function-based Constructive Hyper-Heuristic

This section presents the procedures for initializing the competence and the affinity as well as the Choice Function - based Constructive Algorithm.

A. Competence and Affinity Initialization

In certain conditions, some low level heuristics will behave better than others, and as such should be used more often than low level heuristics that behave more poorly. In order to determine the efficiency of low level heuristics, we use 2 functions: competence and affinity, as presented in []. The competence determines the effectiveness of a low level heuristic when is applied on its own, while the affinity determines the effectiveness of a low level heuristic when is applied after another low level heuristic. In order to compute initial values for both of them (i.e. competence and affinity), we randomly generated set of low level heuristics sequences that are applied on a domain solution randomly generated. From this initial set, we consider a subset(i.e (i.e. a percentage of 20%) that generates the domain solutions with the best fitness.

The algorithms for computing the competence and affinity for the low level heuristics part of the best low level heursitic seqeunces , as presented in [**?**] are presented bellow:

The formula for computing the competence score is defined as:

$$score[h] = \frac{competence[h]}{\sum_{h_i} competence[h_i]} \quad (10)$$

The affinity is computed using this formula:

$$score[h_p, h_i] = \frac{affinity[h_p, h_i]}{\sum_{h1} \sum_{h2} affinity[h1, h2]}$$

Where h_i is the heuristic we want to choose. h_p is the previous heuristic. However, unchecked, these formulas can

Algorithm 1: Competence Initialization Algorithm

Inputs: HL - the set of low level heuristics; HL' - a subset of HL that generate the domain solutions with the best fitness;
Output: $competence[]$ - the competences computed for each heuristics part of a heuristic sequence from the HL' subset;
begin
  foreach hl **in** HL **do**
    $competence[hl] = 0$
  endfor
    foreach hs **in** HL' **do**
      foreach hl' **in** hs **do**
        $competence[hl']++$
      endfor
    endfor
  return $competence[]$;
end

Algorithm 2: Affinity Initialization Algorithm

Inputs: HL - the set of low level heuristics; HL' - a subset of HL that generate the domain solutions with the best fitness;
Output: $affinity[][]$ - the affinities computed for each heuristics part of a heuristic sequence from the HL' subset;
begin
  foreach hl_1 **in** HL **do**
    foreach hl_2 **in** HL **do**
      $affinity[hl_1][hl_2] = 0$
    endfor
  endfor
    foreach hs **in** HL' **do**
      foreach hl'_1 **in** hs **do**
        foreach hl'_2 **in** hs **do**
          $distance = 1/(index(hl_1)' - index(hl'_2))$
          $affinity[hl'_1][hl'_2] += distance$
        endfor
      endfor
  return $affinity[][]$;
end

lead to only a very small number of heuristics being used, while the others are discarded. So, I update the score by dividing it to the time passed since the respective heuristic was used last time. This way, even heuristics that may behave less well will still have their chance to be used.

B. Choice Function-based Constructive Algorithm

The Choice Function-based Constructive Hyper-Heuristic (see Algorithm 1) combines a simulated annealing based strategy with a choice function based strategy to identify the combinations of low-level heuristics that applied on a given domain solution (i.e. daily/weekly menu) assures the converge towards the near optimal/optimal solution. The choice function based strategy takes into consideration (i) how good is a low level heuristics when is applied on a domain solution (i.e. competence)and (ii) how good are combinations of low level heuristics when are applied an a domain solution (i.e. affinity). In our approach, we have considered the following low-level heuristics which can be applied on a domain solution:

- *Single point random mutation* - randomly chooses an element of the domain solution (i.e. food item of the menu) and replaces it with a randomly chosen food item from the set of available food items.
- *Multiple point random mutation* - randomly chooses a number of elements of the domain solution (i.e. food items of the menu) and replaces them with a randomly chosen food items from the set of available food items.
- *Single point memory-based mutation* - randomly chooses an element of the domain solution (i.e. food item of the menu) and replaces it with another food item that appears in the memory of food items.
- *Multiple point memory-based mutation* - randomly chooses a number of elements of the domain solution (i.e. food item of the menu) and replaces them with other food items that appear in the memory of food items.
- *Single point crossover with a randomly generated solution* - performs a crossover between the domain solution and a randomly generated domain solution in a crossover point randomly chosen and outputs the offspring domain solution with the highest fitness value
- *Multiple point crossover with a randomly generated solution* - performs a crossover between the domain solution and a randomly generated domain solution in a number of crossover points randomly chosen and outputs the offspring domain solution with the highest fitness value.
- *Single point crossover with the optimal domain solution* - performs a crossover between the domain solution and the optimal domain solution in a crossover point randomly chosen and outputs the offspring domain solution with the highest fitness value.
- *Multiple point crossover with the optimal domain solution* - performs a crossover between the domain solution and the optimal domain solution in a number of crossover points randomly chosen and outputs the offspring domain solution with the highest fitness value.

The algorithm takes as input the following parameters: HL - the set of low level heuristics hl_i; $alpha$ - a random number used to compute the probability of choosing between competence and affinity; $searchSpace$ - the set of food packages from different providers; $profile$ - the person's profile; per - the percentage of the best low level heuristics sequences that will be initially considered to compute the affinity and the competence; T - the initial temperature; β- the decrease coefficient for temperature; n - the number of randomly generate sequences of low-level heuristics; l - the length of a low level heuristics sequence. The algorithm outputs a daily

food menu that best satisfies the older adult's profile. The algorithm consists of two phases, an initialization phase and an iterative phase. In the initialization phase a number *n* of low-level heuristic sequences are randomly generate and each of them are applied on an initial domain solution randomly generated.

Choose the best heuristic sequences (i.e. 20%), with which the competence and affinity will be initialized using the algorithm described

Generate a random initial solution.

Initialize competence functions

Randomly generate *n* sequences of low-level heuristics

Apply each of heuristic sequences on the initial solution obtained during the previous step.

Choose the best heuristic sequences (i.e. 20%), with which the competence and affinity will be initialized using the algorithm described [subsec:funcs]here.

Initialize sol_{domain} and sol_{opt} with the best solution obtained at the previous step

Initialize the temperature at a high enough value

While the temperature is above a certain threshold:

Pick a low level heuristic either using one of the choice functions (competence or affinity)

Apply the heuristic on the sol_{domain}

If the resulting solution is better than the previous one, update the score components with the current heuristic and set sol_{opt} to the current domain solution

If the resulting solution is worse than the previous one, then compute the possibility of updating the sol_{domain} to the current solution: $P = e^{-\Delta E/T}$, taking values between 0 and 1, where ΔE is the difference between the current fitness and the previous one. This way, we can periodically choose a worse solution in order to escape a local minimum. However, because the fitness function has small values and the temperature can be very high(thousands of degrees), the $e^{-\Delta/T}$is very close to 0, making the probability basically 1 in most iterations. In order to fix this, ΔE should be multiplied by a factor high enough that can lower the overall probability.

Update the Temperature $T = T - \Delta T$

When choosing a low level heuristic to apply on a domain solution, we only use the competence or affinity. We define a constant α between 0 and 1 and then generate a new random sub-unitary number. If this number is smaller than α, we use the competence, otherwise the affinity. Either way, we compute a score for each heuristic, and then choose the heuristic with the best possible score.

III. Results

The proposed Choice Function-based Constructive Hyper-Heuristic Algorithm have evaluated on a set of different user profiles and by considering a search space consisting of a set of 2600 food packages developed in house. The food packages were generated based on a set of food recipes available at []. For the generated food packages the following information is extracted from the set of the food recipes: the name of the food recipe, the categories to which the food recipe belongs (e.g. snack, lunch), the ingredients, and the nutritional values of the food recipes well as. Additional information is generated, for each food package such as the food provider, the cost associated with the food package, the delivery time, and ratings for aspect, taste and smell. Our data set is composed of 98 food packages for breakfast, 100 food packages for snack, 2000 food packages for lunch, and 429 food packages for dinner.

A subset of the user profiles that we used in our experiments are presented in Table I. For each user profile we have considered the following information: the clients name, age, sex, weight in kilograms, height in meters and the associated physical activity factor (i.e. PAF), the preferred price expressed in euros that the client is willing to pay for a menu per day, the preferred time expressed in minutes in which the client wants its menu to be delivered, and the weights given by the client reflecting the degree of importance for the aspect, taste and smell of the food packages.

Based on the user profiles, a nutritionist will compute the nutritional needs corresponding to each user (see Table II). These computed values will be used: (i) by our algorithm when generate personalized menu recommendations and (ii) to analyze the results obtained when evaluating the proposed Chose function based hyperheuristic algorithm.

The adjustable parameters of the Choice Function-based Constructive Hyper-Heuristic Algorithm are the following: $alpha$, T_0 - the initial temperature, and β - the decrease coefficient for temperature. To identify the optimal values of these adjustable parameters, we have performed a set of trial-and-error experiments. This sub-section presents fragments of the best experimental results obtained while tuning the values of the adjustable parameters of the Choice Function-based Constructive Hyper-Heuristic Algorithm for the client profiles and their associated nutritional needs presented in Tables I and II.

For these client profiles we have considered the values illustrated in Tables III for the weights associated to the fitness function components.

Tables IV-**??** presents the top 10 experimental results obtained while varying the values of the Choice Function-based Constructive Hyper-Heuristic Algorithm's adjustable parameters. In our experiments we have sets the following values for the number of randomly generated sequence of low level heuristics, n, the length of a low level heuristic sequence, l, and the percents of the best low level heuristics sequence, per: l=7;....

By analyzing the experimental results we have observed that the choice of the most appropriate configuration of adjustable parameters depends on the compromise willing to be made, whether to obtain a good fitness in a higher execution time, or to obtain a lower good fitness in a low execution time. In the case of our algorithm we have chosen the configuration $T_0 = 85100$, $\beta = 0.9$, and $\alpha = 0.55$ because it provides a good fitness in a lower execution time for all the considered user profiles.

Tables VII- IX presents the recommended menu for the

Algorithm 3: Choice Function-based Constructive Hyper-Heuristic algorithm

Inputs: HL; α; β; $searchSpace$; $profile$; per; T_0;n; l
Output: sol_{opt}
begin
  sol_{init} = **Generate_Random_Domain_Solution**$(searchSpace, profile)$
  $Sol = \{\}$
  $HLLSet$ = **Randomly_Generate_LLHeuristics_Sequences**(HL, n, l)
  foreach in $HLLSet$ **do**
    sol_i = **Apply_Low_Level_Heuristic**(sol_{init})
    Sol = Sol U sol_i
  end foreach
  $HLLSet'$ = **Choose_Best_LLHeuristics_Sequences**$(HLLSet, per)$
  $AffinityScores'$ = **Compute_Affinity(HLLSet')**
  $CompetenceScores'$ = **Compute_Competence(HLLSet')**
  sol_{domain} = **Best_Solution(Sol)**
  sol_{opt} = **Best_Solution(Sol)**
  T = **Initialize_Temperature()**
  while$(T > Threshold)$**do**
    hl_i = **Probabilistic_Pick_LLHeuristic**$(HL, AffinityScores', CompetenceScores', alpha)$
    $sol'_{domain} = sol_{domain}$
    $sol_{domain'}$ = **Apply**$(hl_i, sol_{domain'})$
    if$(Fitness(sol_{domain'}) > Fitness(sol_{domain})$**then**
      $AffinityScores' = Update\_Affinities(HLLSet', hl_i)$
      $CompetenceScores' = Update\_Competences(HLLSet', hl_i)$
      $sol_{domain} = sol'_{domain}$
      $sol_{opt} = sol'_{domain}$
    else
      sol_{domain} = **Probabilistically_Update**(sol'_{domain}, T)
      T = **Update**(T, β)
  return $sol_{domain_{opt}}$
end

TABLE I
CLIENT PROFILES USED IN EXPERIMENTS

ID	Name	Age	Sex	Weight	Height	PAF	Price	Time	Aspect	Taste	Smell
1	John Black	61	M	79	180	1.2	50	175	0.33	0.33	0.33
2	Anne Down	58	F	65	169	1.375	45	120	0.33	0.33	0.33
3	Mary Stan	69	M	78	163	1.55	55	240	0.33	0.33	0.33

TABLE II
CLIENTS' COMPUTED NUTRITIONAL NEEDS

Recommendation	client 1	client 2	client 3
Energy (kcal)	1967.87	1825.46	1653.43
Carbs (g)	221.39 - 319.78	205.37-296.64	186.01 - 206.68
Proteins (g)	49.2 - 172.19	45.64 - 159.73	41.34 - 144.68
Fats (g)	98.39 - 172.19	91.27 - 159.73	82.67 - 144.68
Vit. A (mg)	0.9	0.7	0.7
Vit. B (mg)	1.3	1.3	1.3
Vit. C (mg)	90	75	75
Vit. D (mg)	0.015	0.015	0.015
Calcium (mg)	1200	1200	1200
Iron (mg)	8	8	8
Sodium (mg)	1500	1500	1500

TABLE III
VALUES FOR THE WEIGHTS ASSOCIATED TO THE FITNESS FUNCTION COMPONENTS

Fitness Function Component	**Weight value**
FF_{rec}	$w_1 = 0.5$
FF_{price}	$w_2 = 0.15$
FF_{time}	$w_4 = 0.1$
FF_{rating}	$w_4 = 0.25$

TABLE IV
EXPERIMENTAL RESULTS FOR THE OLDER ADULT PROFILE 1 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM

T_0	β	α	$Fitness_{avg}$	$ExeTime_{avg}$
80700	0.96	0.35	1.39	4.63
92400	0.94	0.3	1.40	4.10
80800	0.92	0.3	1.41	3.31
97600	0.91	0.5	1.46	4.96
89000	0.96	0.45	1.52	2.18
85100	0.9	0.55	1.65	1.47
80200	0.9	0.6	1.67	1.98
85000	0.96	0.55	1.71	1.36
81400	0.96	0.65	1.72	6.19
89100	0.94	0.35	1.83	1.24

three user profiles together with their nutritional values as well as the associated price and the delivery time. The deviations of the results obtained for the nutritional needs recommended by nutritionist for each of the considered user profiles have been introduced by the features of the food packages, as similar to the real life, not all food providers define their food packages according to general nutritional guidelines as well as the low number of available food packages for breakfast.

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

IV. CONCLUSION

The conclusion goes here.

ACKNOWLEDGMENT

The authors would like to thank...

TABLE V
EXPERIMENTAL RESULTS FOR THE OLDER ADULT PROFILE 2 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM

T_0	β	α	$Fitness_{avg}$	$Time_{avg}$
97600	0.91	0.5	1.41	3.72
92400	0.94	0.3	1.44	4.20
85100	0.9	0.55	1.44	1.75
80700	0.96	0.35	1.46	6.16
89100	0.94	0.35	1.47	4.22
80200	0.9	0.6	1.55	3.00
80800	0.92	0.3	1.58	3.29
89000	0.96	0.45	1.61	2.07
81400	0.96	0.65	1.69	1.37
85000	0.96	0.55	1.89	1.36

TABLE VI
EXPERIMENTAL RESULTS FOR THE OLDER ADULT PROFILE 3 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM

T_0	β	α	$Fitness_{avg}$	$Time_{avg}$
85100	0.9	0.55	1.23	3.17
89000	0.96	0.45	1.27	2.02
80700	0.96	0.35	1.28	5.84
80800	0.92	0.3	1.31	2.14
89100	0.94	0.35	1.32	1.28
97600	0.91	0.5	1.33	3.34
92400	0.94	0.3	1.46	2.15
85000	0.96	0.55	1.47	5.78
80200	0.9	0.6	1.67	1.18
81400	0.96	0.65	1.69	1.54

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

V. CONCLUSION

The conclusion goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

TABLE VII
THE RECOMMEND MENU FOR USER PROFILE 1 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM FOR THE BEST CONFIGURATION OF THE ADJUSTABLE PARAMETERS

Nutritional Values	
Proteins(g)(49.20 - 172.19)	63.41
Lipids(g) (98.39 - 172.19)	66.53
Carbohydrates(g) (221.39 - 319.78)	227.79
Energy(kcal) (1967.87)	1731.91
Calcium(mg) (1200.0)	488.44
Iron(mg) (8.0)	9.35
Sodium(mg) (1500.0)	1158.42
Vitamin A(mcg) (900.0)	659.55
Vitamin B6(mg) (1.3)	0.93
Vitamin C(mg) (90.0)	56.12
Vitamin D(mg) (15.0)	1.51
Delivery time and Price	
Price per menu ($)	57.3789
Delivery Time (sec)	243
Recommended Menu	
breakfast	Artichoke and Lettuce Salad
snack 1	Praline Cookies
lunch	Chicken Chowder and Arroz De Sabato
snack 2	Arancia Genovese
dinner	Arni Souvlaki with Armenian Rice Pilaf

TABLE VIII
THE RECOMMEND MENU FOR USER PROFILE 2 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM FOR THE BEST CONFIGURATION OF THE ADJUSTABLE PARAMETERS

Nutritional Values	
Proteins(g)(49.20 - 172.19)	118.23
Lipids(g) (98.39 - 172.19)	70.79
Carbohydrates(g) (221.39 - 319.78)	120.35
Energy(kcal) (1967.87)	1556.85
Calcium(mg) (1200.0)	637.92
Iron(mg) (8.0)	7.46
Sodium(mg) (1500.0)	1689.94
Vitamin A(mcg) (900.0)	608.36
Vitamin B6(mg) (1.3)	0.71
Vitamin C(mg) (90.0)	74.11
Vitamin D(mg) (15.0)	3.32
Delivery time and Price	
Price per menu ($)	55.88872
Delivery Time (sec)	192
Recommended Menu	
breakfast	Artichoke and Lettuce Salad
snack 1	Apple smoothie
lunch	Sweet Potato and Andouille Soup and Arkansas Pecan Chicken Dish with Arroz Con Leche De Coco
snack 2	Nectarine Smoothie
dinner	Acquazzurra

TABLE IX
THE RECOMMEND MENU FOR USER PROFILE 3 WHEN RUNNING THE CHOICE FUNCTION-BASED CONSTRUCTIVE HYPER-HEURISTIC ALGORITHM FOR THE BEST CONFIGURATION OF THE ADJUSTABLE PARAMETERS

Nutritional Values	
Proteins(g)(49.20 - 172.19)	66.72
Lipids(g) (98.39 - 172.19)	66.21
Carbohydrates(g) (221.39 - 319.78)	162.86
Energy(kcal) (1967.87)	1494.60
Calcium(mg) (1200.0)	212.13
Iron(mg) (8.0)	11.66
Sodium(mg) (1500.0)	1580.83
Vitamin A(mcg) (900.0)	495.10
Vitamin B6(mg) (1.3)	0.59
Vitamin C(mg) (90.0)	65.35
Vitamin D(mg) (15.0)	1.70
Delivery time and Price	
Price per menu ($)	43.96
Delivery Time (sec)	155
Recommended Menu	
breakfast	Artichoke and Lettuce Salad
snack 1	Apricot Pie
lunch	Sweet Potato and Andouille Soup and Arcadian Eight Bean Chili
snack 2	Apricot Pear Tart
dinner	Arni Souvlaki with Armenian Rice Pilaf