An Approximate Muscle Guided Global Optimization Algorithm for the Three-Index Assignment Problem

He Jiang, Jifeng Xuan, Xianchao Zhang

***Abstract*—The Three-Index Assignment Problem (AP3) is a famous NP-hard problem with wide applications. Since it's intractable, many heuristics have been proposed to obtain near optimal solutions in reasonable time. In this paper, a new meta-heuristic was proposed for solving the AP3. Firstly, we introduced the conception of muscle (the union of optimal solutions) and proved that it is intractable to obtain the muscle under the assumption that $P \neq NP$. Moreover, we showed that the whole muscle can be approximated by the union of local optimal solutions. Therefore, the Approximate Muscle guided Global Optimization (AMGO) is proposed to solve the AP3. AMGO employs a global optimization strategy to search in a search space reduced by the approximate muscle, which is constructed by a multi-restart scheme. During the global optimization procedure, the running time can be dramatically saved by detecting feasible solutions and extracting poor partial solutions. Extensive experimental results on the standard AP3 benchmark indicated that the new algorithm outperforms the state-of–the-art heuristics in terms of solution quality. Work of this paper not only provides a new meta-heuristic for NP-hard problems, but shows that global optimization can provide promising results in reasonable time, by restricting it to a fairly reduced search space.**

I. Introduction

The Three-Index Assignment Problem (AP3) is a well-known NP-hard problem with wide applications including scheduling capital investments, military troop assignment, satellite coverage optimization, and production of printed circuit boards (Pierskalla, 1967, 1968; Frieze and Yadegar, 1981; Crama, Kolen, Oerlemans and Spieksma, 1990). Since it is intractable for NP-hard problems to obtain optimal solutions, both exact and heuristic algorithms have been developed for solving AP3 instances, including Balas and Saltzman (1991), Burkard and Rudolf (1993), Crama and Spieksma (1992), Pardalos and Pitsoulis (2000), Voss(2000), Aiex, Resende, Pardalos and Toraldo (2005), Huang and Lim (2006). Among these algorithms, LSGA proposed by Huang and Lim can obtain better results than all other existing heuristics.

As an important tool to design heuristics for NP-hard problems, the backbone (i.e., the shared common parts of all optimal solutions for a NP-hard problem instance) has received widespread attention in recent years. Schneider (2003) proposed a multilevel reduction algorithm for the traveling salesman problem (TSP) by using the intersection of local optimal solutions as the approximate backbone. Zhang and Looks (2005) presented a backbone guided LK algorithm for the TSP. Zhang (2004), Dubois and Seymour (2001), and Valnir (2006) gave backbone guided local search algorithms for the SAT, respectively. Zou, Zhou, Chen, Jiang and Gu (2005) developed the approximate backbone-guided fant (ABFANT) for the quadratic assignment problem (QAP).

In contrast to the backbone, the muscle (the union of local optimal solutions) was introduced in this paper. Firstly, we proved that there is no polynomial time algorithm to obtain the muscle of the AP3 under the assumption that $P \neq NP$. Its basic idea is to map any instance of the AP3 to an instance with a unique optimal solution. Therefore, finding the muscle of the modified instance is equivalent to finding an optimal solution of the original instance. Similarly, we indicated that it is intractable to obtain a fixed fraction of the muscle as well. Secondly, we approximated the muscle with the union of local optimal solutions and proposed a new meta-heuristic, the Approximate Muscle guided Global Optimization (AMGO) for the AP3. It consists of two phases: the sampling phase approximates the muscle by running a multi-restart scheme for AP3; the global optimization phase exploits a recursive procedure to find better solutions by restricting the search in the approximate muscle. In the latter phase, the running time can be dramatically reduced in such a way that we detect the feasible solution and extract poor partial solution. Experimental results on the standard AP3 benchmark indicated that the new heuristic outperforms LSGA and GRASP with Path Relinking in terms of solution quality, especially on those hard instances (instances of size > 18 in Balas and Saltzman Dataset).

The rest of this paper is organized as follows. Section II first describes definitions of the AP3 and the muscle, and computational complexity results of the muscle are given. In Section III, we shall approximate the muscle with the union of local optimal solutions, and propose the AMGO heuristic. Experimental results are reported in Section IV. Finally, we conclude this paper in Section V.

II. Muscle of Ap3

A. Preliminaries

In this subsection, we shall give out some useful definitions about the AP3 and the muscle.

Definition 1. Given sets $I = J = K = \{1, 2, \cdots, n\}$ and the cost function $c: I \times J \times K \to \mathbb{R}^+$, where c_{ijk} represents the cost of a triple $(i, j, k) \in I \times J \times K$. A feasible solution to the

This work was supported in part by the National Natural Science Foundation of China under Grant 60673046 and 60673066, the Natural Science Foundation of Liaoning Province under Grant 20051082, and the Gifted Young Foundation of Dalian University of Technology.

He Jiang is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (phone: 86-411-81381830; fax: 86-411-87571567; e-mail: jianghe@dlut.edu.cn).

Jifeng Xuan is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (e-mail: giphon@hotmail.com).

Xianchao Zhang is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (e-mail: xczhang@dlut.edu.cn).

978-1-4244-1823-7/08/$25.00©2008 IEEE

AP3 instance (denoted by $AP3(I,J,K,c)$) is defined as a set $s=\{(i_1,j_1,k_1),(i_2,j_2,k_2),\cdots,(i_n,j_n,k_n)\}$, where $i\neq i'$, $j\neq j'$, $k\neq k'$ for any two distinct triples $(i,j,k),(i',j',k')\in s$.

Definition 2. Given an AP3 instance $AP3(I,J,K,c)$ and a feasible solution s, let $c(s)=\sum_{(i,j,k)\in s} c_{ijk}$ be the cost of s. The AP3 aims to find a feasible solution s^* with minimum cost, i.e., $c(s^*)=\min\{c(s)|s\in\Pi\}$, where Π is the set of all feasible solutions.

Definition 3. Given an AP3 instance $AP3(I,J,K,c)$, let $\Pi^*=\{s_1^*,s_2^*,\cdots,s_q^*\}$ be the set of all optimal solutions to $AP3(I,J,K,c)$, where $|\Pi^*|=q$ represents the number of optimal solutions. The muscle of $AP3(I,J,K,c)$ is defined as $muscle(I,J,K,c)=s_1^*\bigcup s_2^*\bigcup\cdots\bigcup s_q^*$.

It is significant to obtain the muscle $muscle(I,J,K,c)$ for algorithm design. According to definition 3, if the muscle is obtained, the search space could then be effectively reduced by restricting the search to the muscle.

In the following part, we shall introduce some definitions which will be used later.

Definition 4. Given an AP3 instance $AP3(I,J,K,c)$, if there exists exactly one optimal solution to the instance $AP3(I,J,K,c)$, then the instance $AP3(I,J,K,c)$ is a unique optimal solution instance.

Definition 5. Given an AP3 instance $AP3(I,J,K,c)$, the biased AP3 instance is defined as $AP3(I,J,K,\hat{c})$, where $\hat{c}_{ijk}=c_{ijk}+1/2^{in^2+jn+k}$ for every $(i,j,k)\in I\times J\times K$. Given a feasible solution s to $AP3(I,J,K,\hat{c})$, let $\hat{c}(s)=\sum_{(i,j,k)\in s}\hat{c}_{ijk}$ be the cost of s.

Obviously, the biased instance is also an AP3 instance and a feasible solution to the biased instance is also feasible to its original instance. In fact, it only needs $O(n^3)$ running time to construct the biased instance for a given AP3 instance. In the following part, we shall prove that the biased instance has a unique optimal solution, which is also optimal to the original instance.

Definition 6. Given two solutions s_1,s_2 to an AP3 instance $AP3(I,J,K,c)$, the distance between s_1 and s_2 is defined as $dist(s_1,s_2)=n-|s_1\bigcap s_2|$.

B. Computational Complexity of Muscle

Theorem 1. Given an AP3 instance $AP3(I,J,K,c)$, if $c_{ijk}\in\mathbb{Z}^+$ for every $(i,j,k)\in I\times J\times K$, then the biased instance $AP3(I,J,K,\hat{c})$ is a unique optimal solution instance.

Proof. For any two distinct solutions $s_1\neq s_2$ to $AP3(I,J,K,c)$, we only need to verify that $\hat{c}(s_1)\neq\hat{c}(s_2)$.

By definition, we have $\hat{c}(s_1)=c(s_1)+\sum_{(i,j,k)\in s_1}1/2^{in^2+jn+k}$. When viewed as a binary string, $c(s_1)$ will be the integer part, and $\sum_{(i,j,k)\in s_1}1/2^{in^2+jn+k}$ will be the fractional part of $\hat{c}(s_1)$. Thus, the in^2+jn+k bit will be 1 for all $(i,j,k)\in s_1$. Similarly, $\hat{c}(s_2)$ can also be viewed as a binary string.

For any two triples $(i_1,j_1,k_1),(i_2,j_2,k_2)\in I\times J\times K$, unless $i_1=i_2, j_1=j_2, k_1=k_2$, otherwise $1/2^{i_1n^2+j_1n+k_1}\neq 1/2^{i_2n^2+j_2n+k_2}$. Since $s_1\neq s_2$, there must exist one triple (i^*,j^*,k^*) such that $(i^*,j^*,k^*)\in s_1$ and $(i^*,j^*,k^*)\notin s_2$. It implies that the $i^*n^2+j^*n+k^*$ bit of $\sum_{(i,j,k)\in s_1}1/2^{in^2+jn+k}$ will be 1. However, the same bit of $\sum_{(i,j,k)\in s_2}1/2^{in^2+jn+k}$ will be 0. Therefore, we have $\sum_{(i,j,k)\in s_1}1/2^{in^2+jn+k}\neq\sum_{(i,j,k)\in s_2}1/2^{in^2+jn+k}$, i.e., $\hat{c}(s_1)\neq\hat{c}(s_2)$.

Thus, this theorem is proved. □

Lemma 1. Given an AP3 instance $AP3(I,J,K,c)$, if $c_{ijk}\in\mathbb{Z}^+$ for every $(i,j,k)\in I\times J\times K$, then for any two distinct feasible solutions $s_1\neq s_2$, if $c(s_1)<c(s_2)$, then $\hat{c}(s_1)<\hat{c}(s_2)$.

Proof. By assumption of Lemma 1, $c(s_1)<c(s_2)$. Since $c_{ijk}\in\mathbb{Z}^+$ for every $(i,j,k)\in I\times J\times K$, we have $c(s_2)-c(s_1)\geq 1$.

According to the definition of the biased instance, we have $0<\hat{c}(s_1)-c(s_1)=\sum_{(i,j,k)\in s_1}1/2^{in^2+jn+k}<\sum_{(i,j,k)\in s_1}1/2^{n^2}=n/2^{n^2}<1$. Similarly, we have $0<\hat{c}(s_2)-c(s_2)<1$. It implies that $\hat{c}(s_2)-\hat{c}(s_1)=c(s_2)-c(s_1)+(c(\hat{s}_2)-c(s_2))-(c(\hat{s}_1)-c(s_1))>c(s_2)-c(s_1)-1\geq 0$.

Thus, this lemma is proved. □

Theorem 2. Given an AP3 instance $AP3(I,J,K,c)$, if $c_{ijk}\in\mathbb{Z}^+$ for every $(i,j,k)\in I\times J\times K$, then the unique optimal solution to the biased instance $AP3(I,J,K,\hat{c})$ is also optimal to $AP3(I,J,K,c)$.

Proof. By Theorem 1, there exists a unique optimal solution (denoted by s^*) to the biased instance $AP3(I,J,K,\hat{c})$. Obviously, s^* is also a feasible solution to $AP3(I,J,K,c)$. Thus, s^* must be optimal to the AP3 instance $AP3(I,J,K,c)$, otherwise there exists a solution s such that $c(s)<c(s^*)$. However, by Lemma 1, we have $\hat{c}(s)<\hat{c}(s^*)$, a contradiction.

Thus, this theorem is proved. □

Theorem 3. There exists no polynomial time algorithm to obtain the full muscle of the AP3 problem unless $P=NP$.

Proof. Otherwise, there must be a polynomial time algorithm (denoted by Γ) to obtain the muscle of the AP3. A contradiction will be found in the following proof by

constructing a polynomial time algorithm to obtain an optimal solution to the AP3.

Given any arbitrary AP3 instance $AP3(I,J,K,c)$, without loss of generality, we shall assume that $c_{ijk} \in \mathbb{Z}^+$ for every $(i,j,k) \in I \times J \times K$ [1]. We can always obtain an optimal solution to the instance $AP3(I,J,K,c)$ in polynomial time as follows.

Firstly, the instance $AP3(I,J,K,c)$ can be transformed into its biased instance $AP3(I,J,K,\hat{c})$ in $O(n^3)$ running time. Secondly, since the biased instance is also a AP3 instance, its muscle can be obtained by Γ in polynomial time (denoted by $O(\bullet)$). By Theorem 1, the muscle is the unique optimal solution (denoted by s^*) to $AP3(I,J,K,\hat{c})$. In the mean time, by Theorem 2, s^* is also optimal to $AP3(I,J,K,c)$. Therefore, we can always solve the instance $AP3(I,J,K,c)$ in $O(n^3)+O(\bullet)$ running time.

However, it has been proved that no polynomial time algorithm exists to solve a NP-hard problem unless $P = NP$, a contradiction. Thus, this theorem is proved. □

Theorem 4. There exists no polynomial time algorithm to obtain a fixed fraction of the muscle to the AP3 problem unless $P = NP$.

Proof. Otherwise, there must be a polynomial time algorithm (denoted by Λ) to obtain a fixed fraction of the muscle to an AP3 instance. A contradiction will be found in the following proof by constructing an exact algorithm of polynomial time complexity for the AP3 problem.

Given any arbitrary AP3 instance $AP3(I,J,K,c)$, without loss of generality, we shall assume that $c_{ijk} \in \mathbb{Z}^+$ for every $(i,j,k) \in I \times J \times K$. We can always obtain an optimal solution to the instance $AP3(I,J,K,c)$ in polynomial time as follows.

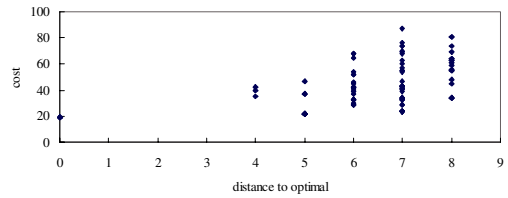
Firstly, the biased instance $AP3(I,J,K,\hat{c})$ can be obtained in $O(n^3)$ running time. By Theorem 1, $AP3(I,J,K,\hat{c})$ has a unique optimal solution s^*, i.e., the muscle. Thus, we can obtain at least one triple (denoted by (i_1,j_1,k_1)) from s^* (the muscle) by Λ in polynomial time (denoted by $p(n)$). In such a way, a new biased instance $AP3(I \setminus \{i_1\}, J \setminus \{j_1\}, K \setminus \{k_1\}, \hat{c})$ can be constructed. Obviously, it is also a unique optimal solution instance from which a new triple (denoted by (i_2,j_2,k_2)) can obtained by Λ in polynomial time (denoted by $p(n-1)$). Then another biased instance $AP3(I \setminus \{i_1,i_2\}, J \setminus \{j_1,j_2\}, K \setminus \{k_1,k_2\}, \hat{c})$ can be constructed.

[1] If some triple costs are decimal fraction, we can simply rescale the instance to a new instance with integer costs only, through multiplying each triple cost by a large number. The optimal solutions to the new instance are identical to the original one.

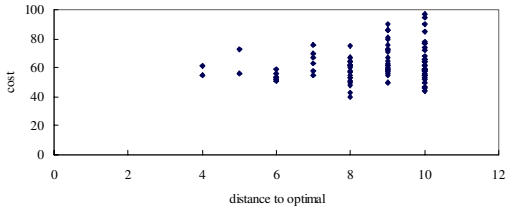
By such sequential $n-1$ calls to Λ, we can obtain an optimal solution $s = \{(i_1,j_1,k_1),(i_2,j_2,k_2),\cdots,(i_n,j_n,k_n)\}$ to $AP3(I,J,K,\hat{c})$. By Theorem 2, this optimal solution is also optimal to $AP3(I,J,K,c)$. Therefore, we can always solve the instance $AP3(I,J,K,c)$ in polynomial running time.

However, it contradicts with the fact that no polynomial time algorithm exists to solve a NP-hard problem under the assumption that $P \neq NP$. Thus, this theorem is proved. □

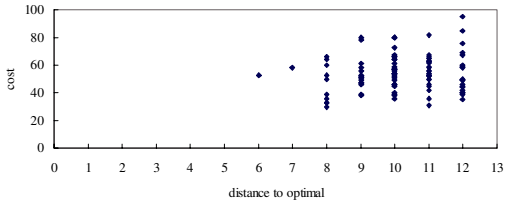
III. Approximate Muscle Guided Global Optimization Algorithm for AP3



(1) bs_8_1.dat



(2) bs_10_1.dat



(3) bs_12_1.dat

Fig. 1: 100 RLS local optimal solutions. Solution cost (vertical axis) is plotted against the distance to the global optimal solution.

A. Approximating the Muscle

As proven in Theorem 3 and Theorem 4, it is intractable to obtain the muscle with any performance guarantee. In this subsection, we shall investigate the way to approximate the muscle.

Boese (1995) observed in the traveling salesman problem (TSP) that a local optimal solution tends to have 80% common edges shared with an optimal solution. Similar phenomena were also found in the graph partitioning problem (Merz and Freisleben, 2000), and the job shop scheduling problem (Reeves, 1999). Such observations have led to the "big valley" structure, which suggests that extensively many local optimal solutions form clusters around the optimal solutions.

Algorithm 1: RLS for AP3
Input: AP3 instance $AP3(I, J, K, c)$
Output: solution s'
Begin
(1) for $i = 1$ to n do $p[i] = i$, $q[i] = i$;
(2) for $i = 1$ to n do
let j be a random integer between 1 and n;
swap $p[i]$ and $p[j]$;
(3) for $i = 1$ to n do
let j be a random integer between 1 and n;
swap $q[i]$ and $q[j]$;
(4) let $s = \{(i, p[i], q[i]) | 1 \le i \le n\}$;
(5) obtain the local optimal solution s' by applying the hungarian local search to s;
End

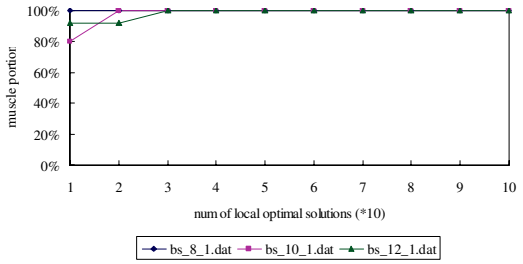


Fig. 2: Muscle portion vs. num of local optimal solutions.

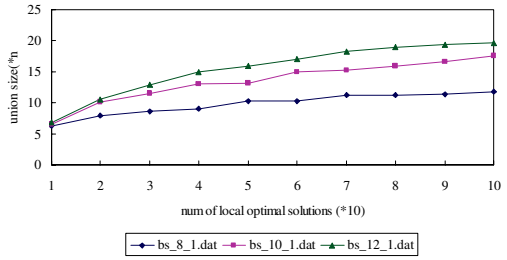


Fig. 3. Union size vs. number of local optimal solutions.

Similar to the work of Boese, we examined the characteristics of local optimal solutions as follows. Firstly, 100 local optimal solutions (denoted by $s_1, s_2, \cdots, s_{100}$)were generated to several typical instances (bs_8_1.dat, bs_10_1.dat, and bs_12_1.dat) from Balas and Saltzman Dataset (Balas and Saltzman, 1991), by 100 runs of a Random Local Search (RLS) algorithm (see Algorithm 1). RLS starts with a randomly generated initial solution (step (1)-step (3)), followed by a hungarian local search (step (4)- step (5)) which was proposed in (Huang and Lim, 2006) by projecting a AP3 instance to a linear assignment problem. Secondly, we computed the distance of each solution to the global optimal solution (see Fig. 1). Our results show a weak relationship between cost and distance in the AP3 problem.

Thus, we conducted a new experiment to investigate the relationship between the muscle and the union of local optimal solutions. As shown in Fig. 2, the union contains almost the whole muscle. When the number of local optimal solutions exceeds 30, the union contains exactly the whole muscle for bs_8_1.dat, bs_10_1.dat, and bs_12_1.dat, respectively. And Fig. 3 shows the growth trend of the union size against the number of local optimal solutions. The union size grows slowly along with the increase of local optimal solutions used. When 100 local optimal solutions are used, the union size will be nearly $11n, 18n, 20n$ for bs_8_1.dat, bs_10_1.dat, and bs_12_1.dat, respectively.

Inspired by the relationship between the union and the muscle, we can then approximate the muscle with the union of local optimal solutions.

Definition 7. Given local optimal solutions $s_1, s_2, \cdots, s_k$ to an AP3 instance $AP3(I, J, K, c)$, an approximate muscle is defined as $a\_muscle(\{s_1, s_2, \cdots, s_k\}) = s_1 \bigcup s_2 \bigcup \cdots \bigcup s_k$.

B. AMGO Algorithm for AP3

Algorithm 2: AMGO for AP3
Input: AP3 instance $AP3(I, J, K, c)$, k, heuristic H
Output: solution s^*
Begin
// sampling phase
(1) $c^* = +\infty$, $s = \varnothing$
(2) for $i = 1$ to k do
(2.1) obtain a solution s_i with H;
(2.2) if $c(s_i) < c^*$ then $c^* = c(s_i)$;
(3) $a\_muscle(\{s_1, s_2, \cdots, s_k\}) = s_1 \bigcup s_2 \bigcup \cdots \bigcup s_k$;

// global optimization phase
(4) for $i = 1$ to n do $fp[i] = false, fq[i] = false$;
(5) obtain s^* with $GO(a\_muscle(\{s_1, s_2, \cdots, s_k\}), s, 1, 0)$;
(6) return s^*;
End

According to the definition, once the muscle is obtained, we can then search in a reduced search space by restricting the search in the muscle. Although it's intractable to obtain the muscle, we have observed in the subsection above that the whole muscle can be approximated by the union of local optimal solutions. Moreover, the approximate muscle size is far smaller than the whole search space. For instance, the approximate muscle size of bs_12_1.dat is nearly $20n$ only, while the total number of triples in bs_12_1.dat will be $n^3 = 144n$. The approximate muscle is so small that a global search can be exploited in the search procedure for obtaining high quality solution in reasonable time. Therefore, we proposed the Approximate Muscle guided Global Optimization (AMGO) for the AP3 problem.

AMGO (see Algorithm 2) is a meta-heuristic which consists of two phases: the sampling phase and the global optimization phase.

(1) Sampling Phase

This phase (step (1)-step (3) in AMGO) aims to sample the search space for constructing the approximate muscle. In the sampling phase, local optimal solutions $s_1, s_2, \cdots, s_k$ are generated by a multi-restart scheme and the best solution cost is recorded by c^*. The approximate muscle is then constructed as the union of $s_1, s_2, \cdots, s_k$. During this phase, any existing heuristic for the AP3 problem can be incorporated into the multi-restart scheme. In this paper, we use RLS as the subordinate heuristic H.

Algorithm 3: GO (Global Optimization)
Input: approximate muscle m_a, initial solution s, i, partial solution cost c_p
Output: solution s^*
Begin
// now a solution is to be completed
(1) if $i = n$ then
  for every triple $(n,k,j) \in m_a$ do
    if $fp[k] = false$ and $fq[j] = false$ then
      if $c_p + c_{nkj} < c^*$ then
      (1.1) $c^* = c_p + c_{nkj}$;
      (1.2) $s^* = s \bigcup \{(n,j,k)\}$;

// a partial solution is constructed
(2) if $i < n$ then
  for every triple $(i,k,j) \in m_a$ do
    if $fp[k] = false$ and $fq[j] = false$ then
      if $c_p + c_{ikj} < c^*$ then
      (2.1) $fp[k] = true$, $fq[j] = true$; *// j,k is fixed*
        // a bigger partial solution is constructed
      (2.2) $s = s \bigcup \{(i,k,j)\}$;
        // GO is called with the new partial solution
      (2.3) $GO(m_a, s, i+1, c_p + c_{ikj})$;
      (2.4) $fp[k] = false$, $fq[j] = false$; *// j,k is freed*
End

(2) Global Optimization Phase

After the approximate muscle is constructed, we can then exploit a global optimization strategy (step (4) – step (5) in AMGO) to a restrained search space. At the beginning of the global optimization phase, the initial solution is set to be empty, and all the elements in J and K of the AP3 instance $AP3(I,J,K,c)$ are free to be used (see step (4) in AMGO). Thus, the Global Optimization (GO) is called to obtain the best solution in the restrained search space.

GO (see Algorithm 3) is a recursive procedure which constructs the solution by adding triples to it one by one. The action of GO can be classified into two cases as follows.

In the first case, a partial solution is to be constructed, i.e., s contains $i-1$ ($(i<n)$) disjoint triples. For every triple $(i,k,j) \in m_a$, two branch-cut actions including feasible solution detection and poor partial solution extraction are conducted. A feasible solution is detected by checking whether k and j have been fixed in s. The poor partial solutions are extracted by ensuring the cost sum ($c_p + c_{ikj}$) of the partial solution and the triple (i,k,j) is less than c^*. After the branch-cut actions, the triple (i,k,j) is joined into s in order to construct a bigger partial solution. GO is then recursively called with the new partial solution (step (2.3) in GO). Before (After) the inner GO is called, the elements j,k are fixed (freed).

In the second case, a full solution is to be constructed. For every triple $(n,k,j) \in m_a$, we check whether k and j haven't been fixed in s and the cost sum $c_p + c_{ikj}$ is less than c^*. If so, a new best solution $s^* = s \bigcup \{(n,j,k)\}$ is completed and the cost c^* of the best solution is refreshed.

C. Aspects of AMGO

As a new meta-heuristic for AP3, AMGO possesses several good aspects as follows.

(1) Flexibility

AMGO provides a framework for NP-hard problems in which any existing heuristic can be employed. In this paper, we just RLS as the subordinate heuristic.

(2) Simplicity

In contrast to the LSGA and GRASP with Path Relinking algorithms for the AP3, the AMGO requires no complex data structure for implementation. And the core code of GO is far simpler than many other operators used in GRASP and LSGA.

(3) Efficiency

Global search is usually a time-exhaustive approach for a NP-hard problem solving, whereas our global search part of AMGO is fairly efficient. Such efficiency comes not only from the approximate muscle, which is far smaller than the total triples of the instance, but from the branch-cut strategies used in GO. With the actions of feasible solution detection and poor partial solution extraction in GO, the running time can be significantly reduced.

Compared to the original solution space, the solution space of GO contains far less solutions. However, the GO can surely obtain feasible solutions due to the fact that solutions $s_1, s_2, \cdots, s_k$ are all belonging to this restrained solution space.

IV. Experimental Results

In this section, we demonstrated the effectiveness of AMGO by experimental results over the AP3 benchmark dataset. All the codes are implemented by C++ under a Pentium IV D2.8GHz with 1G memory. In AMGO, we use 1000 local optimal solutions, in order to ensure the approximate muscle contains as many triples from the muscle as possible in reasonable time.

However, the computing machine in Aiex's paper (2005) is a SGI Challenge R10000 machine, and the computing machine in Huang's paper (2006) is a Pentium III 800. Therefore, in order to compare the CPU time, a scaling

scheme is used according to SPEC (Standard Performance Evaluation Corporation, www.specbench.org/osg/cpu2000/), which indicates that Pentium IV D2.8GHz is not more 14.3 times (3.7 times) faster than SGI Challenge R10000 (Pentium III 800) (see Appendix).

The AP3 dataset is generated by Balas and Saltzman (1991). It includes 60 test instances with the problem size n = 4, 6, 8, ⋯, 26. For each problem size n, five instances are randomly generated with the integer cost coefficients c_{ijk} uniformly distributed in the interval [0, 100].

Tab. 1 shows the results of our experiments on this dataset. Each row reports the average score of the five instances with the same size.

The column "Optimal'' shows the optimal solution reported by Balas and Saltzman, while column "B–S" is the result of their Variable Depth Interchange heuristic. Column "GRASP with Path Relinking" is the result reported in Aiex's paper. Column "LSGA" is the result reported in Huang's Paper. Finally, Column "AMGO" shows our algorithm using the RLS as the subordinate heuristic. The best results among these algorithms are underlined in this table.

It is evident that our AMGO can provide much better solutions than GRASP with Path Relinking. For instances with the problem size <=22, the AMGO uses far less time than GRASP with Path Relinking. The running time of AMGO for instances with the problem size >22 becomes longer due to the global search phase. And AMGO obtains better solution in terms of quality than LSGA as well, especially on instance with size > 18. For instances with the problem size >=16, AMGO spends more time on the global optimization phase.

Tab. 1: Balas and Saltzman Dataset (12*5 instances)

n	Optima	B-S	GRASP with Path Relinking			LSGA			AMGO	
	Avg. Cost	Avg. Cost	Avg. Cost	Avg. CPU time (seconds)		Avg. Cost	Avg.CPU time (seconds)		Avg. Cost	Avg.CPU time (seconds)
				R10000	PIV D2.8G		PIII800	PIV D2.8G		PIV D2.8G
4	42.2	43.2	-	-	-	42.2	0.00	0.00	42.2	0.01
6	40.2	45.4	-	-	-	40.2	0.01	0.003	40.2	0.03
8	23.8	33.6	-	-	-	23.8	0.03	0.008	23.8	0.07
10	19.0	40.8	-	-	-	19.0	0.37	0.1	19.0	0.12
12	15.6	24.0	15.6	74.79	>5.23	15.6	0.87	0.24	15.6	0.21
14	10.0	22.4	10.0	106.55	>7.45	10.0	1.73	0.47	10.0	0.32
16	10.0	25.0	10.2	143.89	>10.06	10.0	1.89	0.51	10.0	0.79
18	6.4	17.6	7.4	190.88	>13.35	7.2	2.95	0.80	6.8	1.77
20	4.8	27.4	6.4	246.70	>17.25	5.2	4.01	1.08	5.0	3.29
22	4.0	18.8	7.8	309.64	>21.65	5.6	4.54	1.23	4.4	8.52
24	1.8	14.0	7.4	382.45	>26.74	3.2	5.66	1.53	2.8	26.48
26	1.3	15.7	8.4	465.20	>32.53	3.6	10.78	2.91	2.4	35.69

V. Conclusion

In this paper, the conception of muscle was firstly introduced as the union of optimal solutions. And we proved that it is intractable to obtain full or part of the muscle. Furthermore, we approximated the muscle with the union of local optimal solutions. With the approximate muscle, the AMGO was then proposed to solve the AP3 problem. The new meta-heuristic applies a restrained global optimization strategy to a fairly reduced search space, after the approximate muscle is constructed by a multi-restart sampling phase. Extensively experimental results on open benchmarks indicated that the AMGO had achieved dramatic improvement over existing heuristics.

Work of this paper provides a case study of theoretical analysis for the muscle of NP-hard problems. Similar skills may be applied to other NP-hard problems. Moreover, the AMGO provides a new meta-heuristic for other NP-hard problems. Our work also throws a light on the use of global optimization in NP-hard problems. Global optimization used to be so time-consuming that it is seldom used in heuristics, whereas this paper shows that global optimization can provide promising results in reasonable time as well, by restricting it to a fairly reduced search space.

Appendix

According to Tab. 2, Pentium IV D2.8G: R10000 = (1424 / 233) * (20.7 /8.85) = 14.2949 < 14.3; Pentium IV D2.8G: Pentium III800= 1424 / 386 = 3.6891 < 3.7.

Tab. 2: CPU Benchmark from SPEC

	Intel PIV D2.8G	Intel P III800	Intel PIII 500	SGI Challenge R1000
SPECint 95			20.7	8.85
SPECint 2000	1424	386	233	

REFERENCES

[1] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, G. Toraldo, "GRASP with Path Relinking for Three-Index Assignment," INFORMS Journal on Computing, 17(2), pp. 224-247, 2005.

[2] E. Balas, M.J. Saltzman, "An algorithm for the three-index assignment problem," Operations Research vol.39, pp. 150-161, 1991.

[3] K.D. Boese, "Cost versus distance in the traveling salesman problem," Technical Report CSD-950018, 1995.

[4] R.E. Burkard, R. Rudolf, "Three dimensional axial assignment problems with decomposable cost coefficients," Discrete Applied Mathematics vol.32, pp. 85-98, 1993.

[5] Y. Crama, A.W.J. Kolen, A.G.Oerlemans, F.C.R.Spieksma, "Throughput rate optimization in the automated assembly of printed circuit boards," Annals Operations Research, vol.26, pp. 455-480, 1990.

[6] Y. Crama, F.C.R. Spieksma, "Approximation algorithms for three-dimensional assignment problems with triangle inequalities," European Journal of Operations Research vol.60, pp. 273-279, 1992.

[7] O. Dubois, P. Seymour, "A backbone-search heuristic for efficient solving of hard 3-SAT formula," in Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), San Francisco: Morgan Kaufmann Publishers, pp. 248-253, 2001.

[8] A.M. Frieze, J. Yadegar, "An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice," Journal of Operations Research Society, vol.32, pp. 989-995, 1981.

[9] P. Hansen, L. Kaufman, "A primal-dual algorithm for the three-dimensional assignment problem," Cahiers du CERO vol.15, pp. 327-336, 1973.

[10] G. Huang, A. Lim, "A hybrid genetic algorithm for the Three-Index Assignment Problem," European Journal of Operational Research, 172(1), pp.249-257, 2006.

[11] P. Kilby, J. Slaney, T. Walsh, "The backbone of the traveling salesperson," in Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI05), San Francisco: Morgan Kaufmann Publishers, pp. 175-181, 2005.

[12] P. Merz, B. Freisleben, "Fitness landscapes and memetic algorithms and greedy operators for graph bi-partitioning," Evolutionary Computation, 8(1), pp. 61-91, 2000.

[13] P.M. Pardalos, L.S. Pitsoulis, "Nonlinear assignment problems: algorithms and applications," Kluwer Academic Publishers, Boston, MA, 2000

[14] W.P. Pierskalla, "The tri-substitution method for the three-dimensional assignment problem," Canadian Operations Research Society Journal, vol.5, pp. 71-81, 1967.

[15] W.P. Pierskalla, "The multidimensional assignment problem," Operations Research, vol.16, pp. 422-431, 1968.

[16] C.R. Reeves, "Landscapes, operators and heuristic search," Annals of Operation Research, 86(1), pp. 473-490, 1999.

[17] J. Schneider, "Searching for backbones-a high-performance parallel algorithm for solving combinatorial optimization problems," Future Generation Computer Systems, 19(1), pp.121-131, 2003.

[18] F. J. Valnir, "Backbone guided dynamic local search for propositional satisfiability," in Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI & Math -06), New York: Springer, pp. 100-108, 2006.

[19] S. Voss, "Heuristics for nonlinear assignment problems," P.M. Pardalos, L.S. Pitsoulis, eds. Nonlinear Assignment Problems: algorithms and applications. Kluwer Academic Publishers, Boston, MA, pp. 175-215, 2000.

[20] W.X. Zhang, "Configuration landscape analysis and backbone guided local search: Part I: satisifiability and maximum satisfiability, " Artificial Intelligence 158(1), pp.1-26, 2004.

[21] W.X. Zhang, M. Looks, "A novel local search algorithm for the traveling salesman problem that exploit backbones," in Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), San Francisco: Morgan Kaufmann Publishers, pp. 343-351, 2005.

[22] P. Zou, Z. Zhou, G.L. Chen, H. Jiang, J. Gu, "Approximate-backbone guided fast ant algorithms to QAP," Journal of Software 16(10), pp. 1691-1698, 2005.