Smooth Constrained Heuristic Optimization of a Combinatorial Chemical Space

by Berend Christopher Rinderspacher

Approved for public release; distribution is unlimited.
NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
Smooth Constrained Heuristic Optimization of a Combinatorial Chemical Space

by Berend Christopher Rinderspacher

*Weapons and Materials Directorate, ARL*
May 2015
Final

Smooth Constrained Heuristic Optimization of a Combinatorial Chemical Space

Berend Christopher Rinderspacher

US Army Research Laboratory
ATTN: RDRL-WMM-G
Aberdeen Proving Ground, MD 21005

ARL-TR-7294

Approved for public release; distribution is unlimited.

primary author’s email: <berend.c.rinderspacher.civ@mail.mil>.

Several algorithms for optimizing a combinatorial subspace of chemical compound space with constraints are compared. The test system is a library of organic chromophores for electro-optic applications. The hyperpolarizability is maximized while the absorption in the range of 400–700 nm is constrained to zero. The best payoff in terms of primary objective, feasibility, and computational cost is achieved using a heuristic reordering of orthogonal search directions.

chemical compound space, discrete optimization, nonlinear optics
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Optimization Methodology</td>
<td>3</td>
</tr>
<tr>
<td>3. Computational Chemistry Protocol</td>
<td>7</td>
</tr>
<tr>
<td>4. Results and Discussion</td>
<td>8</td>
</tr>
<tr>
<td>5. Conclusions</td>
<td>11</td>
</tr>
<tr>
<td>6. References</td>
<td>12</td>
</tr>
<tr>
<td>Appendix. Listings</td>
<td>17</td>
</tr>
<tr>
<td>List of Symbols, Abbreviations, and Acronyms</td>
<td>31</td>
</tr>
<tr>
<td>Distribution List</td>
<td>32</td>
</tr>
</tbody>
</table>
List of Figures

Fig. 1  Optimization framework: Each X may be replaced by -H, -F, -Cl, or -Br
for a total of $2^{16}$ possible molecules. ..................................................4

Fig. 2  Flowchart of algorithm.................................................................6

Fig. 3  Best candidate found .................................................................8
List of Tables

Table 1  Comparison of the performance of the assorted reordering schemes on multiple starting compounds ......................................................8
Table 2  Ratio of objective to constraint, rounded to uniquely identifying ratios 10
Table 3  Comparison of average performance .......................................... 10
1. Introduction

Inverse molecular design (IMD) beyond the purview of drug design has enjoyed increasing popularity recently. Examples can be found in protein design\textsuperscript{1–4} or high-hyperpolarizability materials.\textsuperscript{5–7} The design problem is complicated by the vastness of possible chemicals, termed chemical space. This space can be viewed as combinatorially complex (e.g., $20^8 \approx 2.6 \cdot 10^{10}$ octapeptides of the naturally occurring amino acids exist alone). As a consequence, a variety of methods have been developed for the discrete optimization in chemical subspaces.\textsuperscript{8–15} The continuous optimization of chemicals used in the linear-combination-of-atomic-potentials (LCAP) method\textsuperscript{16,17} and the variation-of-particles density-functional-theoretical (VP-DFT) method introduces an important concept for dealing with the inherent roughness of chemical compound space.\textsuperscript{18–20} LCAP and VP-DFT interpolate continuously between the Hamiltonians of various chemical species. Furthermore, an investigation into the reasons why chemical optimization is, relatively speaking, “easy” recently utilized probability distributions and expectations on the control variables to arrive at its conclusion.\textsuperscript{21,22} Follow-up work discovered that indeed all properties can be optimized efficiently using only the charge density with as-yet unknown functionals.\textsuperscript{23} All previous optimizations were executed using at most single constraints. In this contribution, we introduce a flexible method for multiple nonlinear (inequality) constraints applied to a combinatorially complex search space of energetic materials.

In recent years, organic molecules have garnered increasing attention as components of high-hyperpolarizability materials, partly due to the variety of synthetically accessible compounds, cost, and ease of processing.\textsuperscript{24,25} Applications for materials with high hyperpolarizabilities are found in telecommunication and optics.\textsuperscript{26} The dominant nonlinear response of organic molecules often finds its origin in the conjugated $\pi$-system, which facilitates the electronic polarizability. This lends itself to advanced electronic and photonic applications including optical information processing, photovoltaic cells, photodynamic therapy agents, and many others.\textsuperscript{26–28} The design of such molecules \textit{in silico} is complicated by the fact that chemical space, even constrained to smaller organic compounds, is combinatorially complex. The number of organic molecules of medium size is estimated to be on the order of $10^{200}$.\textsuperscript{29} Enumeration is therefore unfeasibly costly and other methods for property optimization need to be developed. Including conformational searching further complicates molecular design.
The optical spectra and nonlinear optical (NLO) properties of chromophores may be conveniently manipulated by modifying the molecular architecture, substitutional groups, or substitution patterns. The hyperpolarizability tensor are the third derivatives of the energy with respect to the electric field.

\[ \beta_{ijk} = \frac{\partial^3 E}{\partial F_i \partial F_j \partial F_k}. \] (1)

Commonly only the averaged vector components of the vector, \( \beta_i = \frac{1}{3} \sum_j \beta_{ijj} + \beta_{iji} + \beta_{jji} \) are used. In particular the component in the direction of the dipole moment, \( \beta_\mu = \frac{1}{||\mu||} \sum_{i=x,y,z} \beta_i \mu_i \), and the vector norm, \( \beta^2_0 = \sum_{i=x,y,z} \beta^2_i \), play an important role in electro-optic activity.

Several equivalent methods predict electro-optic trends correctly, despite the abundance of methods and conventions for determining hyperpolarizabilities experimentally. The sum-over-states expression of the tensor components in the limit of infinite wavelength,

\[ \beta_{ijk} = \sum_{\mu,\nu \neq 0} \frac{\langle 0|x_i|\mu\rangle \langle \mu|x_j|\nu\rangle \langle \nu|x_k|0\rangle}{E^\alpha_0 E^\nu_0}, \] (2)

where \( E^\alpha_0 \) is the excitation energy from the ground state to excited state \( \alpha \), links the linear absorption spectrum to the hyperpolarizability. From Eq. 2, low excitation energies coupled with large ground-to-excited-state transition-dipole-moments lead to large hyperpolarizabilities (i.e., a material which absorbs everywhere exhibits maximal hyperpolarizability). But such a material would not be able to transmit anything. Hence, it is necessary to navigate the trade-off between hyperpolarizability and optical constraints by tuning the excited-to-excited-state transition-dipole-moments while maintaining transparency or absorption at the target wavelength(s).

Therefore, the main challenge to designing efficient, electro-optic (EO) materials lies in retaining good linear optical properties. For instance, transparent chromophores in a visible yellow spectral range are typically small molecules with low EO response, while molecules possessing large hyperpolarizabilities are frequently opaque or, at best, have small windows of visibility. The visibility window in these NLO frameworks is generally bracketed by 2 kinds of electronic transitions. The blue transition is a local excitation of \( \pi-\pi^* \) character, while
the red transition is characterized by a significant intramolecular charge-transfer. The charge-transfer peak is typically the maximum absorption peak ($\lambda_{max}$) while the $\pi-\pi^*$ transition has a lower oscillator strength and is often denoted as $\lambda_{max-1}$. Optimal chromophores, which are transparent in the visible spectrum, would have a red $\lambda_{max}$-peak, while the $\lambda_{max-1}$-peak should lie in the blue or UV spectral region to open a large visibility window. In the case of telecommunications applications, it is important to achieve transparency in the 1.30–1.55 $\mu$m minimum range. Therefore, the design principle for such chromophores calls for the $\lambda_{max}$-peak to be significantly blue-shifted. In this report, we investigate the effects due to the electronic properties of the $\pi$-system of the bridge between donor and acceptor.

Computational tools are increasingly playing a role in assisting experimental design of optimal chromophores by curtailing the vast optimization space of chromophore structures. Density-functional theory currently offers the best compromise between accuracy and computational performance for typical chromophores of about 100 atoms. The electronic spectra are often calculated using time-dependent density-functional theory (TD-DFT) and numerous successes of this technique have been recently reviewed.

2. **Optimization Methodology**

Our optimization problem is formulated as a constrained maximization,

$$\max_{x \in CCS} P(x)$$

s.t. : $\pi(x) = \sum \epsilon_i^{(x)} \int_{\epsilon_i^{(x)}/400}^{\epsilon_i^{(x)}/700} f_i^{(x)} e^{-\sigma(y-1)^2} dy \leq 30$,

where $\pi(x)$ is the penalty due to the absorbance of compound $x$ from the chosen subspace (CCS) of chemical compound space discussed later in the range 400–700 nm, $\epsilon_i^{(x)}$ and $f_i^{(x)}$ are the excitation wavelength and the normalized oscillator strength of state $i$ for compound $x$, respectively, $\sigma = 72$ is an approximate broadening coefficient, and $P(x)$ is the hyperpolarizability $\beta_0$. The constraint value of 30 is equivalent to an average absorption of 0.1 in the 400- to 700-nm range. We reformulate Problem 3 via a non-negative Lagrange multiplier $0 \leq \lambda \in \mathbb{R}$ for the constraint in the augmented Lagrangian function.
\[ \mathcal{L}(x, \lambda) := P(x) - \lambda \pi(x) \]

\[
\max_{x \in CCS} \min_{\lambda \geq 0} P(x) - \lambda \pi(x). \tag{4}
\]

The Problem 4 is then solved alternatingly between the primal variable(s) \( x \) and the dual variables \( \lambda \).

We begin by describing the optimization procedure within CCS. The chemical subspace that we investigate is depicted in Fig. 1. Each of the 8 Xs on 1 represents a substitution site for which one of 4 chemical groups is attached.

1

**Fig. 1** Optimization framework: Each X may be replaced by -H, -F, -Cl, or -Br for a total of \( 2^{16} \) possible molecules.

The overall search space thus encompasses \( 2^{16} \) molecules. Each such substitution site represents an independent search direction.\(^\text{14}\) There is a natural enumeration that follows from the input (see Listing 4 in the Appendix), where the compound \( \{x_i = X_i\} \) is assigned the number \( n(x) = \sum_i \prod_{j=0}^{i-1} N_j X_i \), where \( N_j \) is the number of options on substitution site \( j \) and \( N_0 = 1 \).

The substitutions are enumerated and thus can be viewed as the integer positions on a circle. The optimization proceeds with a local line search for each substitution site in the prescribed order of sites. When neighboring substitutions in the current direction to the current iterate are inferior, the line search is halted and the next direction is searched.

After all search directions have been searched, the minimization with respect to \( \lambda \) is performed using the dual function \( P_{(d)}(\lambda) = \max_{x \in CCS} \mathcal{L}(x, \lambda) \) via its approximation

\[
P'_{(d)}(\lambda) = \max_{x \in CCS'} \mathcal{L}(x, \lambda) \leq P_{(d)}(\lambda), \tag{5}
\]

where \( CCS' \) is the subset of visited (i.e., already computed) molecules. Due to the discrete nature of \( CCS' \), \( P'_{(d)} \) is a piece-wise linear function of \( \lambda \) with general
derivative \( -\pi(x(\lambda)) \), where 
\[
x(\lambda) := \arg\max_{y \in CCS'} \mathcal{L}(y, \lambda).
\]
Due to the piece-wise linearity, \( \lambda \) only changes meaningfully when 
\( x(\lambda) \neq x(\lambda') \). Thus, whenever \( \pi(x) \neq 0 \), \( \lambda \) is updated cumulatively by
\[
\Delta \lambda = \alpha \pi(x^*) \max \left\{ \inf \{ \lambda' \in \mathbb{R}_+ | \mathcal{L}(x, \lambda^* + \lambda') > \mathcal{L}(x^*, \lambda^* + \lambda') \} , 0 \right\},
\]
(6) where \( \alpha > 1 \), \( CCS' \) is the set of molecules that have already been computed, \( x^* \) is the currently active molecule, and \( \lambda^* \) is the currently active Lagrange multiplier.

By choosing the update along the constraint violation direction \( \pi(x^*) \), the update of \( \lambda \) is a steepest descent update with a conservative step size. The step size is chosen such that either \( x^* \) violates constraints the least as well as maximizing \( P \) or that there exists \( x' \in CCS' \) with an improved \( \mathcal{L}(x', \lambda + \Delta \lambda) > \mathcal{L}(x^*, \lambda + \Delta \lambda) \). The optimization then returns to optimizing over CCS as described before until no improvement can be attained or the maximum number of steps is exceeded. The general program flow is visualized in Fig. 2.

Since generally there is no \textit{a priori} knowledge of the proper search order of each substituent, the initial assignment is expected to be generally unsuited for smooth optimization. To mitigate this problem, the enumeration of substituents is reassessed after each full cycle of local searches in each direction. For each substituent the average arc tangent of the Lagrangian value is computed,
\[
\pi_i^{(j)} = \begin{cases} 
\frac{\sum_{x \in CCS'_{|x_i = j}} \arctan \mathcal{L}(x, \lambda)}{\# \{ x \in CCS'_{|x_i = j} \} }, & \exists x \in CCS' : x_i = j, \\
\alpha_i^j, & \text{otherwise}
\end{cases}
\]
(7) where \( \alpha_i^j \) is a default policy for unseen substituents (discussed on the next page), which is subsequently used to order the substituents around the circle starting with the lowest scoring substituent and placing substituents alternatingly to the left and right in ascending order. The resulting order produces a smooth ordering which is monotonically increasing until the maximum is reached, and then monotonically decreasing until the minimum is reached, or vice versa depending on which part of the circle one starts and which direction one goes. The arc tangent is used because some computations may fail due to convergence issues in geometries or general instability of the molecule, which are mapped to a Lagrangian value of \( -\infty \).

We investigate 4 different strategies for assigning values to unseen substituents.
Algorithm 1: Complete a full sweep of all substituents on the first iteration. After the initial scan there are no unseen substituents anymore.

Algorithm 2: Assign the unseen substituent the same value as the current iterate; this guarantees that the unseen substituents are immediate neighbors of the current iterate and that at least one will be seen in the next optimization step.

Algorithm 3: Assign the unseen substituent the value 0. This is easily implemented but may have random results.

Algorithm 4: Assign the unseen substituents the maximum value. This implies a bias toward unseen substituents to be better in general. As the optimization proceeds this converges with the second method.
No reordering is denoted as Algorithm 0. Four further algorithms are compared:

- **Algorithm 5**: Conduct a full line search in each direction. No reordering is necessary, but presumably far more compounds need to be computed for each direction.

- **Algorithm 6**: Reordering occurs as per Algorithm 3, but unlike in Eq. 6, \( \lambda \) is chosen such that \( \mathcal{L}(\lambda, x^*) \leq \mathcal{L}(\lambda, x') \), where \( x' \in \arg \min_{y \in \text{CCS}'} \pi(y) \). Hence, Algorithm 6 solves the subproblem

\[
\begin{align*}
\max_{x \in \text{CCS}'} & \quad P(x) \\
\text{s.t.} & \quad \pi(x) \leq 30.
\end{align*}
\]

- **Algorithm 7**: Following the protocol of optimizing on a hypercube.\(^{14}\)

- **Algorithm 8**: Same as Algorithm 7, but with deleting compounds \( x \) with \( \pi(x) > \pi(x^*) \) from the library, effectively reordering the library, where \( x^* \) is the current iterate.

Since the latter 2 methods are unaware of the underlying structure of the search space, they may be expected to perform erratically.

### 3. Computational Chemistry Protocol

All geometries were optimized with PM6 using *Gaussian 09*\(^{46}\) as described elsewhere\(^{47}\) (see Listing 1 in the Appendix for specifics). Hyperpolarizabilities \( \beta_\mu \) were computed at the CNDO level of theory.\(^{14}\) Spectra were computed at the BH&HLYP/6-31+G(d) level in the time-dependent density-functional framework\(^{48,49}\) using *Gaussian 09*.\(^{46}\)

Before further analysis, the conformational space was explored to ensure only low-energy species were considered.\(^{47}\) Listing 2 in the Appendix summarizes these steps.
4. Results and Discussion

Despite expectations Algorithm 5 does not always perform better at reducing the violation than the other algorithms (see start number 41668 in Table 1). As expected, the Algorithm 7 variants tend to perform worse than the Algorithm 5 variants. Surprisingly, the Algorithm 8 actually performs worse on average than Algorithm 7. Algorithm 6 uses the fewest iterations on average per individual run. The largest number of computed compounds in a single run was 138, which constitutes merely 0.02% of the library. Despite minimizing $\pi$, Algorithm 6 results in the lowest penalty violation once (see start number 41668 in Table 1). Except for start number 8389, Algorithm 0 is inferior with respect to fulfilling constraints to all other methods but Algorithm 8, which fails to outperform Algorithm 0 for start number 0 only.

Table 1 Comparison of the performance of the assorted reordering schemes on multiple starting compounds

<table>
<thead>
<tr>
<th>Method</th>
<th>Start</th>
<th># Molecules</th>
<th>Property</th>
<th>Penalty</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5</td>
<td>0</td>
<td>119</td>
<td>121.31</td>
<td>15.5295</td>
<td>0.000541122</td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>85</td>
<td>133.88</td>
<td>23.2389</td>
<td>0.000506251</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>81</td>
<td>124.352</td>
<td>21.5181</td>
<td>0.000573735</td>
<td></td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>74</td>
<td>133.88</td>
<td>23.2389</td>
<td>0.000581504</td>
<td></td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>74</td>
<td>124.352</td>
<td>21.5181</td>
<td>0.000628007</td>
<td></td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>93</td>
<td>126.962</td>
<td>18.8063</td>
<td>0.00057176</td>
<td></td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>49</td>
<td>124.352</td>
<td>21.5181</td>
<td>0.000948418</td>
<td></td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>64</td>
<td>132.119</td>
<td>23.5097</td>
<td>0.000664619</td>
<td></td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>80</td>
<td>130.764</td>
<td>27.0287</td>
<td>0.000462471</td>
<td></td>
</tr>
<tr>
<td>Algorithm 5</td>
<td>4710</td>
<td>114</td>
<td>121.31</td>
<td>15.5295</td>
<td>0.000564856</td>
</tr>
<tr>
<td>Method</td>
<td>Start #</td>
<td>Molecules</td>
<td>Property</td>
<td>Penalty</td>
<td>Efficiency</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-----------</td>
<td>----------</td>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>45</td>
<td>164.923</td>
<td>38.6927</td>
<td>0.000574326</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>84</td>
<td>139.494</td>
<td>21.5761</td>
<td>0.000551757</td>
<td></td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>91</td>
<td>121.31</td>
<td>15.5295</td>
<td>0.000707622</td>
<td></td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>93</td>
<td>121.31</td>
<td>15.5295</td>
<td>0.000692404</td>
<td></td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>93</td>
<td>121.31</td>
<td>15.5295</td>
<td>0.000692404</td>
<td></td>
</tr>
<tr>
<td>Algorithm 5</td>
<td>76</td>
<td>125.011</td>
<td>17.7595</td>
<td>0.000740893</td>
<td></td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>55</td>
<td>125.011</td>
<td>17.7595</td>
<td>0.00102378</td>
<td></td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>48</td>
<td>163.287</td>
<td>35.6982</td>
<td>0.000583596</td>
<td></td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>8389</td>
<td>120</td>
<td>3.5658</td>
<td>0.002337016</td>
<td></td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>111</td>
<td>105.054</td>
<td>3.5658</td>
<td>0.002526504</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>118</td>
<td>105.054</td>
<td>3.5658</td>
<td>0.002376627</td>
<td></td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>139</td>
<td>105.339</td>
<td>8.68666</td>
<td>0.000828195</td>
<td></td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>139</td>
<td>105.333</td>
<td>8.68666</td>
<td>0.000828195</td>
<td></td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>128</td>
<td>125.799</td>
<td>18.8994</td>
<td>0.000413373</td>
<td></td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>63</td>
<td>93.8483</td>
<td>5.66247</td>
<td>0.002803196</td>
<td></td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>64</td>
<td>104.418</td>
<td>8.92695</td>
<td>0.001750318</td>
<td></td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>81</td>
<td>105.333</td>
<td>8.68666</td>
<td>0.001421223</td>
<td></td>
</tr>
<tr>
<td>Algorithm 5</td>
<td>41329</td>
<td>142</td>
<td>3.5658</td>
<td>0.001974943</td>
<td></td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>75</td>
<td>164.923</td>
<td>38.6927</td>
<td>0.000344596</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>111</td>
<td>112.876</td>
<td>11.5523</td>
<td>0.000776385</td>
<td></td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>111</td>
<td>111.944</td>
<td>7.44892</td>
<td>0.001209438</td>
<td></td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>112</td>
<td>105.054</td>
<td>3.5658</td>
<td>0.002503946</td>
<td></td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>111</td>
<td>111.944</td>
<td>7.44892</td>
<td>0.001209438</td>
<td></td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>85</td>
<td>96.982</td>
<td>5.35516</td>
<td>0.002196892</td>
<td></td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>77</td>
<td>129.505</td>
<td>23.4977</td>
<td>0.000552693</td>
<td></td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>97</td>
<td>125.799</td>
<td>18.8994</td>
<td>0.000545482</td>
<td></td>
</tr>
<tr>
<td>Algorithm 5</td>
<td>41668</td>
<td>107</td>
<td>115.973</td>
<td>12.729</td>
<td>0.000734213</td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>88</td>
<td>132.798</td>
<td>22.3781</td>
<td>0.000507802</td>
<td></td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>104</td>
<td>93.8483</td>
<td>5.66247</td>
<td>0.00169809</td>
<td></td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>137</td>
<td>126.962</td>
<td>18.8063</td>
<td>0.000388129</td>
<td></td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>87</td>
<td>115.973</td>
<td>12.729</td>
<td>0.000902997</td>
<td></td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>138</td>
<td>126.962</td>
<td>18.8063</td>
<td>0.000385316</td>
<td></td>
</tr>
</tbody>
</table>
Table 1 Comparison of the performance of the assorted reordering schemes on multiple starting compounds (continued)

<table>
<thead>
<tr>
<th>Method</th>
<th>Start</th>
<th># Molecules</th>
<th>Property</th>
<th>Penalty</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 6</td>
<td>101</td>
<td>105.054</td>
<td>3.5658</td>
<td>0.002776653</td>
<td></td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>87</td>
<td>118.986</td>
<td>13.5325</td>
<td>0.000849381</td>
<td></td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>112</td>
<td>105.333</td>
<td>8.68666</td>
<td>0.001037109</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Ratio of objective to constraint, rounded to uniquely identifying ratios

<table>
<thead>
<tr>
<th>Method</th>
<th>0</th>
<th>4710</th>
<th>8389</th>
<th>41329</th>
<th>41668</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5</td>
<td>7.81</td>
<td>7.81</td>
<td>29.46</td>
<td>29.46</td>
<td>9.11</td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>5.76</td>
<td>4.26</td>
<td>29.46</td>
<td>4.26</td>
<td>5.93</td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>5.78</td>
<td>6.47</td>
<td>29.46</td>
<td>9.77</td>
<td>16.57</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>5.76</td>
<td>7.81</td>
<td>12.13</td>
<td>15.03</td>
<td>6.75</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>5.78</td>
<td>7.81</td>
<td>12.13</td>
<td>29.46</td>
<td>9.11</td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>6.75</td>
<td>7.81</td>
<td>6.66</td>
<td>15.03</td>
<td>6.75</td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>5.78</td>
<td>7.04</td>
<td>16.57</td>
<td>18.11</td>
<td>29.46</td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>5.62</td>
<td>7.04</td>
<td>11.66</td>
<td>5.51</td>
<td>8.79</td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>4.84</td>
<td>4.57</td>
<td>12.13</td>
<td>6.66</td>
<td>12.13</td>
</tr>
</tbody>
</table>

Table 3 Comparison of average performance

<table>
<thead>
<tr>
<th>Method</th>
<th>Total # Molecules</th>
<th>Total Efficiency</th>
<th>Average Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 5</td>
<td>602</td>
<td>0.00046585</td>
<td>0.00123043</td>
</tr>
<tr>
<td>Algorithm 0</td>
<td>404</td>
<td>0.000694163</td>
<td>0.000891896</td>
</tr>
<tr>
<td>Algorithm 1</td>
<td>498</td>
<td>0.000563136</td>
<td>0.001196011</td>
</tr>
<tr>
<td>Algorithm 2</td>
<td>552</td>
<td>0.000243202</td>
<td>0.000742978</td>
</tr>
<tr>
<td>Algorithm 3</td>
<td>505</td>
<td>0.000555331</td>
<td>0.00111111</td>
</tr>
<tr>
<td>Algorithm 4</td>
<td>563</td>
<td>0.000238451</td>
<td>0.000654458</td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>374</td>
<td>0.000749845</td>
<td>0.00189321</td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>347</td>
<td>0.000322825</td>
<td>0.000968158</td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>418</td>
<td>0.000275404</td>
<td>0.000809976</td>
</tr>
</tbody>
</table>
5. Conclusions

We have developed and compared efficient algorithms for handling constraints in the optimization of substitutional subspaces of chemical compound space. The best candidate was found to be 2 (see Fig. 3). Algorithm 6 shows the most promise for finding good candidates with minimal effort both for single runs as well as aggregate samples. Reordering substitutions improves expected performance for single runs as both Algorithm 1 and Algorithm 3 are very competitive with Algorithm 6. Since Algorithm 6 differs from Algorithm 3 merely in solving the subproblem, there is no additional computational cost associated. Algorithm 4 shows the worst performance both in aggregate as on average due to relegating entire subspaces maximally far from the current iterate.

The Algorithm 7 family of algorithms performs surprisingly well on average, even though it does not utilize the underlying structure of the search space. On the other hand, aggregate efficiency is poor for the Algorithm 7 family.

Since Algorithm 7 consistently computes the fewest compounds before converging, it can be recommended when only a quick assessment is required. In all other cases, Algorithm 6 should be considered superior.
6. References


Appendix. Listings
Listing 1 energy_run.sh runs the constrained geometry preoptimization and the follow-up full optimization. It returns the final coordinates.

```bash
#!/bin/bash
# Execution script
# remove extra files
EXEC=$PWD/g09_run
filename=`basename $1 .dat`
mkdir -p $1

if [ ! -e "$1/energy" -a ! -e "$1/failed" ]; then
cd $1
    echo %chk=opt.chk > pre.com
    cat ../../header.com >> pre.com
    cat ../../$1.zmat >> pre.com
    echo %chk=opt.chk > opt.com
    cat ../../footer.com >> opt.com
    if [ ! -e "pre.log" ]; then
        NORMALEXEC=`tail pre.log | grep -o termination`
        if [ ! -z "$NORMALEXEC" -o ! -e "opt.chk" ]; then
            $EXEC pre.com pre.log
        fi
    fi
$EXEC opt.com opt.g09_out
NORMALEXEC=`tail opt.g09_out | grep -o Normal | awk '{ print $1 }'
if [ "$NORMALEXEC" == "Normal" ]; then
    fgrep opt.g09_out -e 'SCF Done' | tail -l | awk '{ print $5 }' > energy
    awk -f ../../logcart.awk < opt.g09_out > xyz
    python ~/bin/retrieve_zmat_from_xyz.py ../$1.zmat xyz
    | grep c > opt.rcons
    python ~/bin/retrieve_zmat_from_xyz.py ../$1.zmat xyz
    | grep d > opt.rvars
    ../retrieve_zmat_g03 opt.g09_out opt
```
Listing 2 proprty_script.sh computes the properties and penalties of a molecule.

```bash
#!/bin/bash
# Execution script
# geometry optimize in Gaussian and feed to CNDO
# run cndo
# read strongest excitation
# read beta
# remove extra files
EXEC=/apps/gaussian/scripts/g09l_run_d01
filename=`basename ${1} .dat`
mkdir -p $1

if [ ! -e "$1/$1.failed" -a ! -e "$1/$1.xyz" ]; then
cd $1
# Compute Hyperpolarizability
if [ ! -e "$1/result" ]; then
# CNDO
../xyz_to_CNDO $1.xyz $1.dat 0 1
ln -s ../SOS_input.txt .
ln -s ../IND01S.par .
echo $1 | cndo >& cndo_out
SOSx
../clean_cndo
```

19
mv sos* $1.sos
fgrep $1.sos -e 'beta mu' | awk '{ print $3}' > $1.result
BLA='cat $1.result`
../fabs $BLA > $1.result
bzip2 $1.log
cd ..
ln -s $1/$1.result.
cd $1
fi
if [ ! -e "$1.penalty" ]; then
bunzip2 $1.chk.bz2

# Compute penalty of spectrum

echo %chk=$1.chk > $1_spectrum.com
cat ../spectrum.com >> $1_spectrum.com
$EXEC $1_spectrum.com $1_spectrum.log 32
bzip2 $1.chk
STATEG09=’fgrep $1_spectrum.log -e Normal | awk \`
' { print $1}'
if [ "$STATEG09" == "Normal" ]; then
fgrep $1_spectrum.log -e "Excited State" | 
awk '{ print $7}' > $1.nm
fgrep $1_spectrum.log -e "Excited State" | 
awk '{ print $9}' | awk -F = '{ print $2}' > $1.osc
../integrate_window_penalty ${1}.nm $1.osc 
${1}.penalty
BLA='cat $1.penalty`
../fabs $BLA > $1.penalty

rm $1.chk.bz2
bzip2 *log
else
echo 100000 > $1.penalty
touch $1.failed
exit 1
Listing 3 Script that sets up the right links for the optimization after a good initial structure has been identified.

```bash
#!/bin/bash

n=$1
echo $n
ln -s $1 $2
for i in `ls ${n}.* | awk -F "${n}" '{print $2}'`; do
  # echo $i
  if [ ! -e "$2$i" ]; then
    ln -s $1$i $2$i
  fi
done
n=$1
cd $1
for i in `ls ${n}*chk* | awk -F "$n" '{print $2}'`; do
cp $1$i $2$i
done
for i in `ls ${n}.*.* | awk -F "$n" '{print $2}'`; do
  # echo $i
  if [ ! -e "$2$i" ]; then
    ln -s $1$i $2$i
  fi
done
cd ..
```

Listing 4 Input file to determine the substitutional search space.

ChemGroup((
Z( # Framework
(N ,−3 , 0.000000 , −2 , 0.000000 ,−1 , 0.000000)
(C , 0 , 1.455198 , −3 , 0.000000 , −2, 0.000000)
(C , 0 , 1.455202 , 1 ,118.635000 ,−3 , 0.000000)
(C , 0 , 1.382367 , 1 ,119.869000 , 2 , 165.442000)
(H , 1 , 1.090564 , 0 ,109.068000 , 3 ,−177.996000)
(H , 1 , 1.100788 , 0 ,112.488000 , 3 , 61.924000)
(H , 1 , 1.096133 , 0 ,111.145000 , 3 ,−59.556000)
(H , 2 , 1.090564 , 0 ,109.068000 , 3 , 178.005000)
(H , 2 , 1.100785 , 0 ,112.487000 , 3 ,−61.916000)
(H , 2 , 1.096133 , 0 ,111.145000 , 3 , 59.565000)
(C , 3 , 1.418413 , 0 ,121.400000 , 2 , 172.824000)
(C ,10 , 1.387462 , 3 ,121.166000 , 0 , 178.587000)
(C ,11 , 1.410705 , 10 ,121.553000 , 3 , 0.387000)
(C ,12 , 1.410706 , 11 ,117.355000 ,10 , 0.288000)
(C ,13 , 1.387460 , 12 ,121.552000 ,11 ,−0.288000)
(C ,12 , 4.060996 , 11 ,121.324000 ,10 ,−179.837000)
(C ,15 , 1.413713 , 12 ,120.840000 ,11 , 0.091000)
(C ,16 , 1.387454 , 15 ,120.846000 ,12 , 179.824000)

22
( 0, 0, 1)
( 0, 0, 1)
( 0, 0, 0)
()
)
(
(26, 21, 37)
(0.7, 0, 0)
(0, 120, 0)
(0, 0, 0)
( 0, 0, 1)
( 0, 0, 1)
( 0, 0, 0)
()
)
)
)
allowed_groups(
(1, 2, 3, 4)  # Subs, H, F, Cl, Br
(1, 2, 3, 4)
(1, 2, 3, 4)
(1, 2, 3, 4)
(1, 2, 3, 4)
(1, 2, 3, 4)
(1, 2, 3, 4)
(1, 2, 3, 4)
)
)

(#1
Z(
( H, −3, 0.45, −2, 0, −1, 0)
)
ReturnConnector()
Connector()
allowed_groups()
Listing 5 integrate_window_penalty.cc Computes the penalty function for absorbing in the visible electromagnetic range.

#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <cmath>

#define MAX_STEP 20
using namespace std;

double tolerance=1e-14;

double num_int(double x, double y, double alpha)
{
    double d=y-x;
    double dx=d/100000.0;
    double r=0.0;
    for(long i=0; i<100001; i++, x+=dx)
        r+=dx*exp(-(1.0-x)*(1.0-x)/(x*x)*alpha*alpha);
    return r;
}

double diff_int_Gaussian(double x, double y)
{
    long i;
    double z=1;
    double r=0.0;
    const double x2=x*x;
    const double x4=x2*x2;
    const double y2=y*y;
    const double y4=y2*y2;
    double x4i=x;
    double y4i=y;
    double r1=0.0;
    double r2=0.0;

    for(i=0; i<MAX_STEP && z>tolerance; i++)
    {
        z=1.0/(double)(4*i+1)*(y4i-x4i)-
        (y2*y4i-x2*x4i)/(double)((4*i+3)*(2*i+1));
        r1 +=1.0/(double)(4*i+1)*(x4i)-
        (x2*x4i)/(double)((4*i+3)*(2*i+1));
        r2 +=1.0/(double)(4*i+1)*(y4i)-
        (y2*y4i)/(double)((4*i+3)*(2*i+1));
    }
r += z;
x4i* = x4/(double) ((2*i+1)*(2*i+2));
y4i* = y4/(double) ((2*i+1)*(2*i+2));
}
return r2-r1;
}

int main(int argc, char *argv[])
{
  if (argc < 3)
    cerr << "Usage: " << argv[0]
        << " <nmfile> <oscfile> <NLO_optresultfile> \n";
  ifstream log(argv[1]);
  ifstream log2(argv[2]);
  fstream result;

  bool done=false;
  string search;

  int number;
  double eV, cm_1, nm;
  double strength;
  double penalty = -30.0;
  log >> nm;
  log2 >> strength;
  double alpha = sqrt(72.0);
  while (log.good() && !done) {
    penalty += strength * num_int(400.0/nm, 700.0/nm, alpha)*nm;
    log >> nm;
    log2 >> strength;
  }
  if (penalty < 0) penalty = 0.0;
  result.open(argv[3], ios::in);
  getline(result, search);
}
result.close();
stringstream ss2;
double val = 0.0;
ss2 << search;
ss2 >> val;
val -= penalty;
result.open(argv[3], ios::out);
result << fabs(val) << endl;
result.flush();
result.close();
<table>
<thead>
<tr>
<th><strong>List of Symbols, Abbreviations, and Acronyms</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CCS</strong></td>
</tr>
<tr>
<td><strong>EO</strong></td>
</tr>
<tr>
<td><strong>LCAP</strong></td>
</tr>
<tr>
<td><strong>NLO</strong></td>
</tr>
<tr>
<td><strong>nm</strong></td>
</tr>
<tr>
<td><strong>TD-DFT</strong></td>
</tr>
<tr>
<td><strong>VP-DFT</strong></td>
</tr>
</tbody>
</table>